1993

Nonlinear system identification and prediction

Manu K. Mathew

NONLINEAR SYSTEM IDENTIFICATION AND PREDICTION

by

Manu K. Mathew

A Thesis Submitted
in
Partial Fulfillment
of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Mechanical Engineering

Approved by:

J.S. Torok - Thesis Advisor

Prof. C. W. Haines (Department Head)

Prof. R. Orr (Mathematics Department)

Department of Mechanical Engineering
College of Engineering
Rochester Institute of Technology
Rochester, New York
1993

NONLINEAR SYSTEM IDENTIFICATION AND PREDICTION

I, Manu K. Mathew, hereby grant permission to the Wallace Memorial Library of RIT, to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

December 15, 1993.

"The forecast," said Mr. Oliver, turning the pages till he found it, "says: Variable winds; fair average temperature, rain at times."..... There was a fecklessness, a lack of symmetry and order in the clouds, as they thinned and thickened. Was it their own law, or no law, they obeyed?

-- VIRGINIA WOOLFE, Between the Acts.

Everybody talks about the weather; it's one thing we have in common. On a given afternoon, sunshine may fall on our porch while a resident in another part of town may have falling rain. But weather is the phenomenon we share. With its variability, general dependability, and moment to moment unpredictability, weather infiltrates our schedules, sets or undermines our plans, affects our moods, and unites us with the environment and each other. Weather is only one of the thousands of examples of the mysterious influence of chaos.

ACKNOWLEDGMENTS

For helping me write and complete this thesis, I would like to take this opportunity to graciously thank the following people:

- Dr. Joseph Torok, for all the help he has given me in this endeavor, for pushing me to find, and understand more about the subject, and for the support and guidance he gave me during my school years to cope with the madness and chaos of each quarter.

- Edward Zeigler, for all the times he has gone out of his way to help me out in my time of need, and for the numerous occasions he has helped me to solve many problems during the past year.

- Soni, my fiancée, for all her support in my work, for always being there, and especially for her patience during the endless hours I spent either writing or working in the lab.

- My Dad and Mom, without whom, I would not have this cherished chance for a great future, and for all the sacrifices they have made to make all my dreams within grasp.

Thank you all

Manu.

TABLE OF CONTENTS

	PAGE
1. Introduction	1
2. Creating the Delayed Vector Space	
(i) Finding the optimal decorrelation value	12
(ii) Determining the embedding dimension	27
3. Constructing a Nonlinear Model	35
4. Prediction using developed Nonlinear Model	50
5. Conclusion	60
6. Appendices	63
7. References	88

INTRODUCTION

The great power of science lies in its ability to relate cause and effect. On the basis of the laws of gravitation, for example, eclipses can be predicted thousands of years in advance. There are other natural phenomenon that are not as predictable. The weather, the flow of a mountain stream, the roll of the dice all have unpredictable aspects. Since there are is no clear relation between cause and effect, such phenomena are said to have random elements.

Yet until recently, there was little reason to doubt that precise predictability could in principle be achieved. It was assumed that it was only necessary to gather and process a sufficient amount of information. Such a viewpoint has been altered by a striking discovery: simple deterministic systems with only a few elements can generate apparent random behavior. The randomness is fundamental; gathering more information does not make it go away. Randomness generated in this way has come to be called chaos.

A seeming paradox is that chaos is deterministic, generated by fixed rules that do not themselves involve any elements of chance. In principle, the future is completely determined by the past, but in practice small uncertainties are amplified, so that even though the behavior is predictable in the short term, it is unpredictable in the long term. *There is an order in chaos, generating chaotic behavior that creates randomness in the same way as a card dealer shuffles a deck of cards or a blender mixes cake batter.*

Chaos is the irregular behavior of simple deterministic equations. Irregular fluctuations are ubiquitous in both natural and artificial systems. Chaos is an

appealing notion to a physicist confronted with a dynamical system that exhibits aperiodic fluctuations, because it implies that these fluctuations might be explained with relatively simple deterministic equations, then it becomes possible to predict future fluctuations, at least in the short term, and perhaps even to control them.

Traditionally, the modeling of a dynamical system proceeds along one of two lines. In one approach, deterministic equations of motion are derived from principles, initial conditions are measured, and the equations of motion are integrated forward in time. Alternatively, when a model is unavailable or intractable, then the dynamics can be modeled as a random process, using non deterministic and typically linear laws of motion.

Until recently, the notions of determinism and randomness were seen as opposites and were studied as separate subjects with little or no overlap. Complicated phenomena were assumed to result from complicated physics among many degrees of freedom, and thus were analyzed as random processes. Simple dynamical systems were assumed to produce simple phenomena, so only simple phenomena were modeled deterministically.

Chaos provides a link between deterministic systems and random processes, with both good and bad implications for the predication problem. In a deterministic system, chaotic dynamics can amplify small differences, which in the long run produces effectively unpredictable behavior. On the other hand, chaos implies that not all random-looking behavior is the product of complicated physics. Under the intoxicating influence of nonlinearity, only a few degrees of freedom

are necessary to generate chaotic motion. In this case, it is possible to model the behavior deterministically and to make short-term predictions that are far better than those that would be obtained from a linear stochastic model. Chaos is thus a double-edged sword; it implies that even though approximate long-term predictions may be impossible, very accurate short-term predictions may be possible.

Apparent randomness in time series may be due to chaotic behavior of a nonlinear but deterministic system. In such cases it is possible to exploit the determinism to make short-term forecasts that are much more accurate than one could make from a linear stochastic model. This is done by first reconstructing a state space and then using nonlinear regression methods to create a dynamical model. Nonlinear models are valuable not only as short-term forecasters, but also as diagnostic tools for identifying and quantifying low-dimensional chaotic behavior.
Building a dynamical model directly from the data involves two steps:

1. *State space reconstruction.* In general, we can define the phase space or state space, as the space whose axes are the coordinates of position and velocity, and the phase trajectory, as a curve in this space representing the evolution of the system. A state $\boldsymbol{x(t)}$ is an information set, typically a real vector, which fully describes the system at a fixed time t. If it is known with complete accuracy and if the system is strictly deterministic, then the state contains sufficient information to determine the future of the system. The goal of the state space reconstruction is to use the immediate past behavior of the time series to reconstruct the current state of the system, at least to a level of accuracy permitted by the presence of noise.
2. *Nonlinear function approximation.* The dynamics can be represented by a function that maps the current state $\boldsymbol{x(t)}$ to a future state $\boldsymbol{x(t + T)}$. An approximation to the dynamics can be found by fitting a nonlinear function to the graph of all pairs

of the form $(x(t),x(t+T))$. Note that in order to model chaotic behavior, the functional form of f must be nonlinear.

The systems being presented are the following nonlinear systems, which are best described using the equations that follow [16] :

ROSSLER :

$$X' = -(Y + Z)$$
$$Y' = X + 0.2\,Y$$
$$Z' = 0.4 - 5.8\,Z + X\,Z$$

LORENZ :

$$X' = -10\,X + 10\,Y$$
$$Y' = 28\,X - Y - X\,Z$$
$$Z' = -2.67\,Z + X\,Z$$

The systems described above will be used to test the robustness of the methods being used in this study.

(These systems were implemented using Matlab, through the programs 'rossler.m' and 'lorenz.m'; Appendix 5 & 6 respectively). Some examples of "state spaces" also known as "phase portraits" are described on the following pages (Figures 1-5).

The purpose of this investigation, was to study modeling techniques used to describe a non linear system and the implementation of the chaos theory to determine the level of predictability that these models provide with regards to the actual behavior of the system. To summarize briefly the following steps are involved in this modeling process:

(1) **Developing a delayed vector space using an obtained time series.**

(The concept of delayed vector spaces will be elaborated on later in more detail)

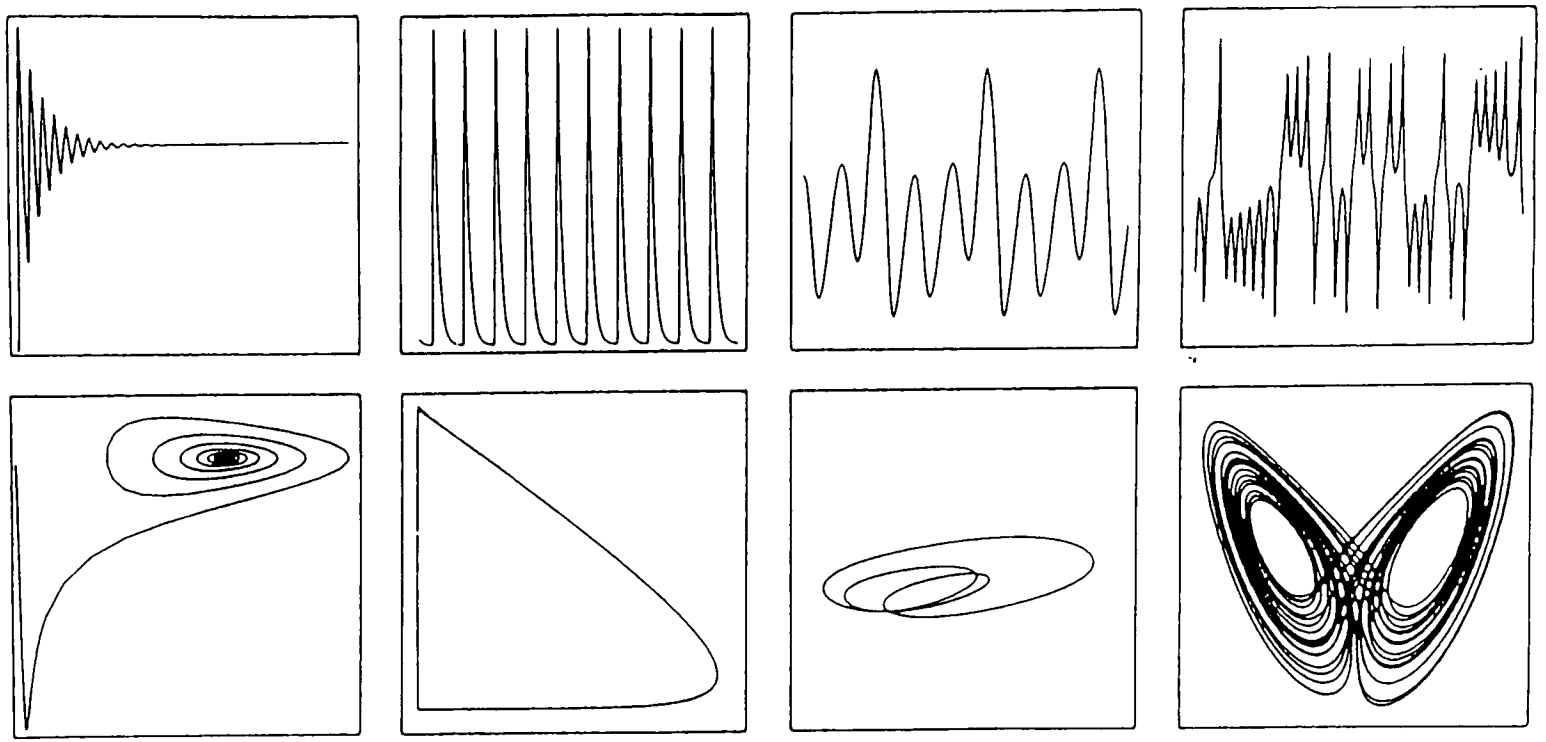
(i) This requires finding the delay time τ.

(ii) It also means that the dimensionality of the system would have to be determined. i.e. the minimum number of degrees of freedom required to describe the system.

(2) **The next step is to derive a set of dynamical equations to best approximate the system.**

(3) **Finally we need to study the predictability of the system based on the derived equations.**

PORTRAITS IN PHASE SPACE
FIGURE 1



Traditional time series (above) and trajectories in phase space (below) are two ways of displaying the same data and gaining a picture of a system's long-term behavior. The first system (left) converges on a steady state- a point in phase space. The second repeats itself periodically, forming a cyclical orbit. The third repeats itself in a more complex waltz rhythm, a cycle with "period three." The fourth system exhibts chaotic behavior.

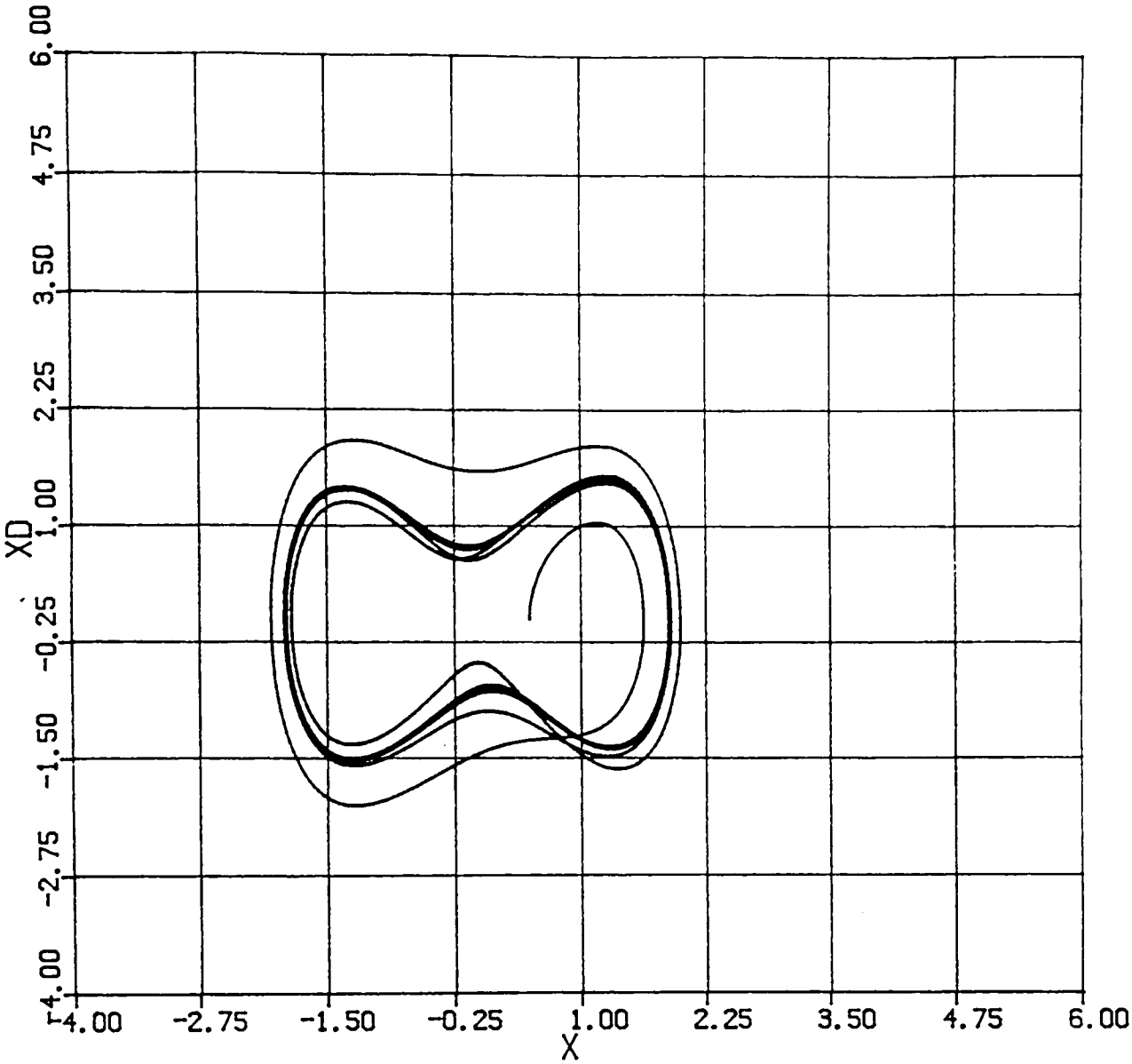


FIGURE 2

***The Duffing equation*; a mechanical system with a nonlinear spring is governed by:**

$$\ddot{x} + k\dot{x} + ax + bx^3 = A\cos\omega t$$

where k is the damping constant related to the system

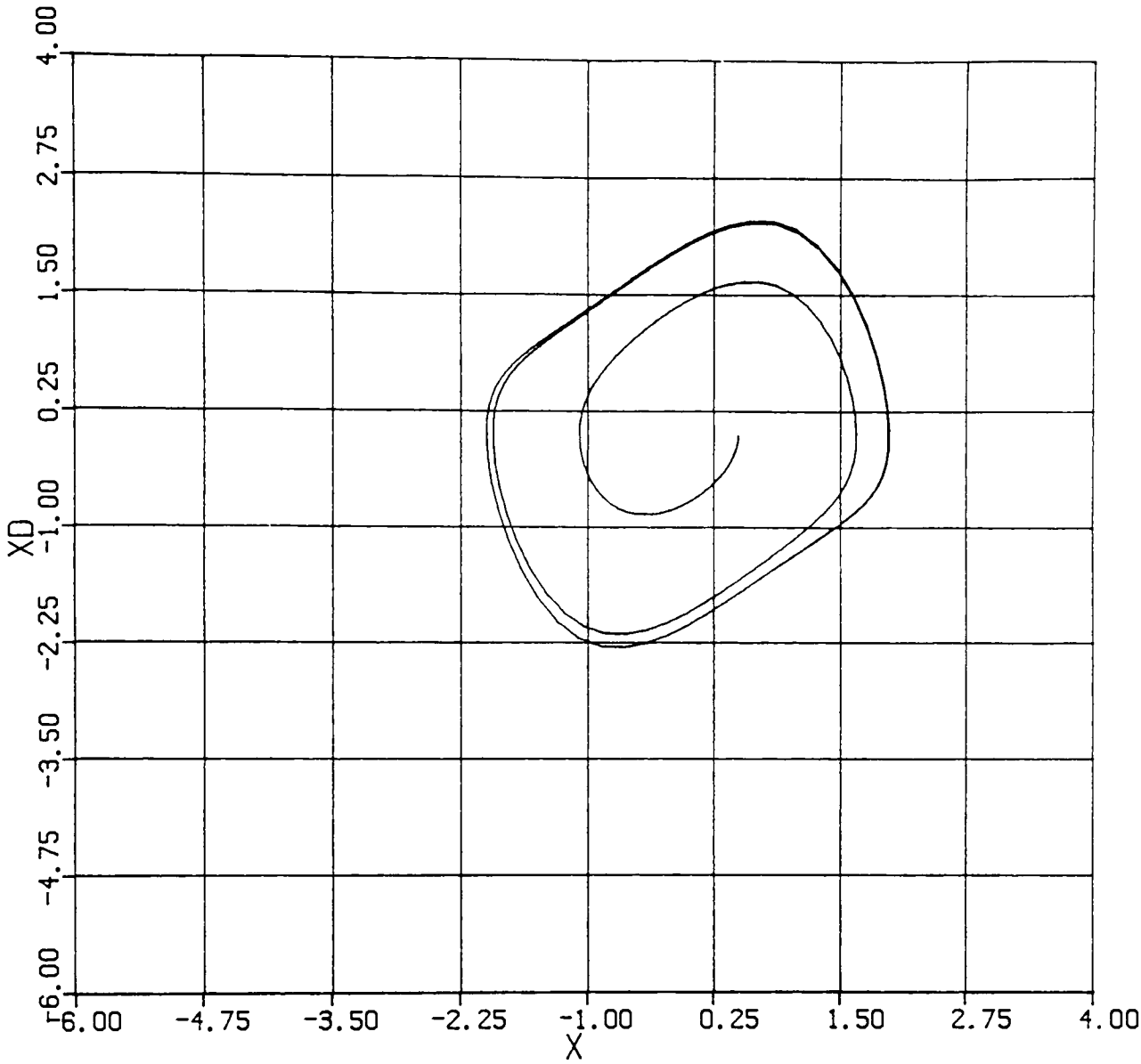


FIGURE 3

***The Van der Pol Oscillator*; an analysis of a non-linear oscillator which contains a non-linear damping term which causes self excited oscillation. The differential equation is expressed below:**

$$\ddot{x} - k(a - 3bx^2)\dot{x} + x = A\sin\omega t$$

where k is the damping term related to the system.

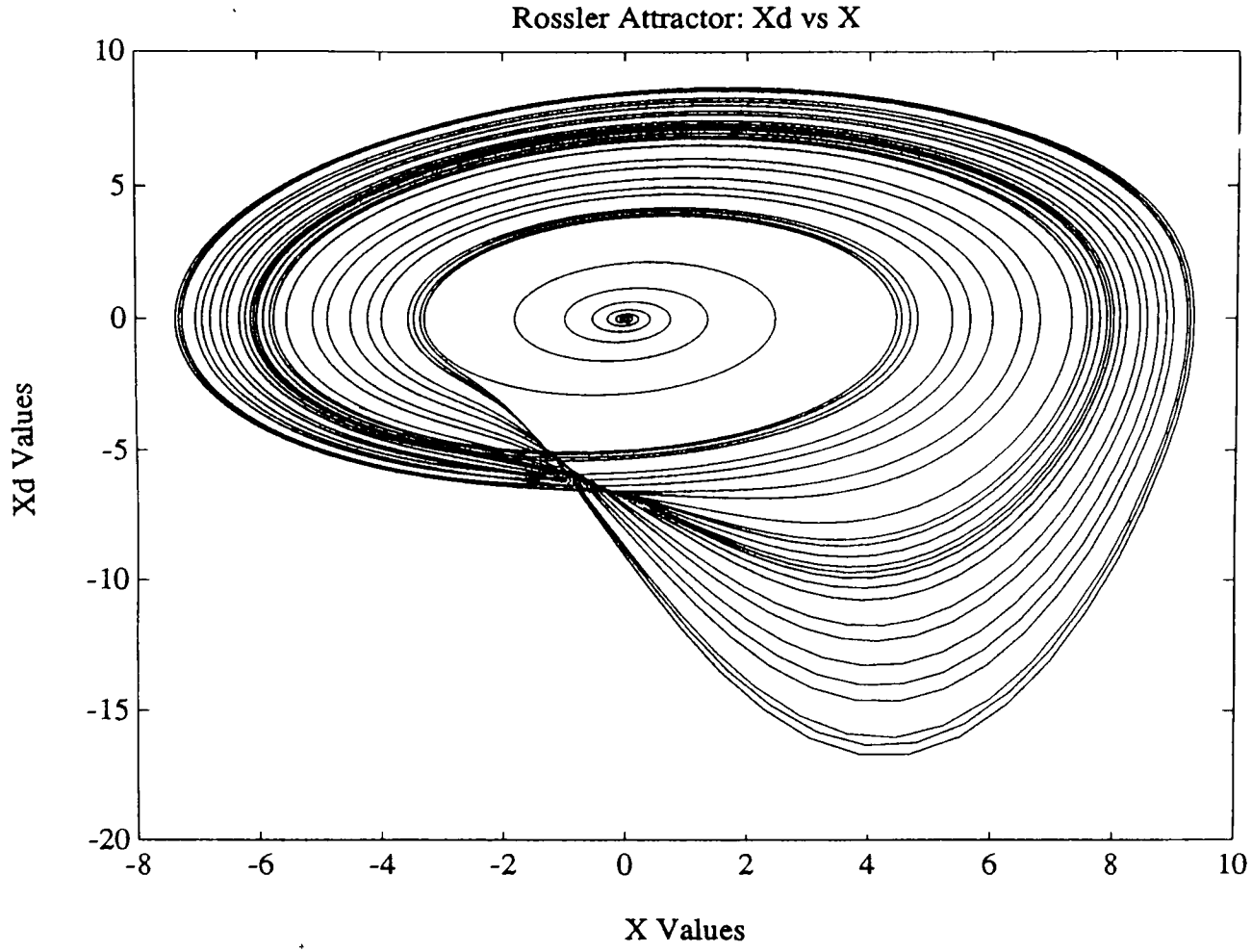


FIGURE 4

A View of the Rossler Strange Attractor: Volcanic eruptions from a volcano in the Philipipines were accompanied by tremors which, when plotted, revealed this underlying strange attractor [21]. Thus the Rossler Attractor plots the transition from order to chaos, but also the transition from chaos to order.

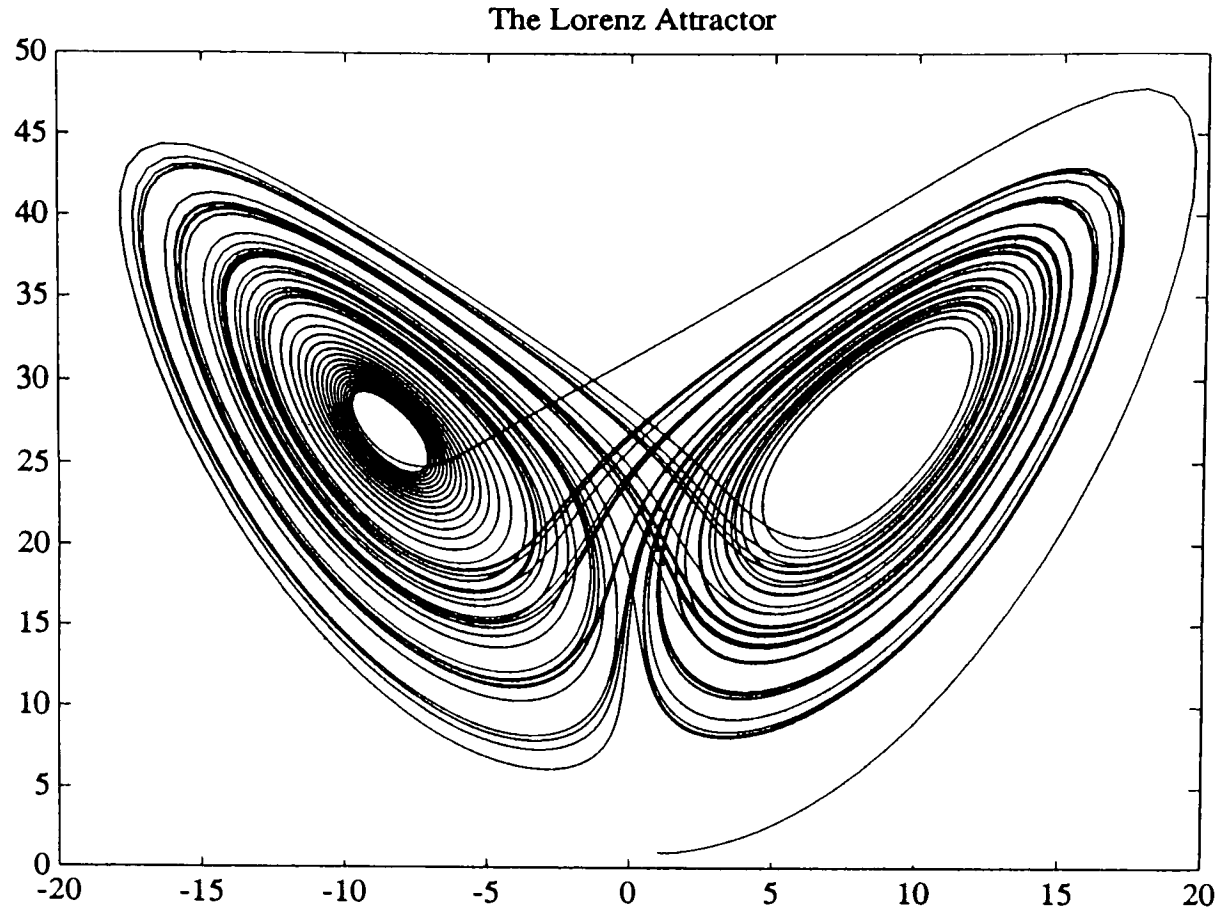


FIGURE 5

This is another face of chaos, the butterfly mask of unpredictablity discovered in the early 1960's by one of the first chaologists Edward Lorenz. He obtained this attractor through studies involving the weather system. Strange attractors like this one depict a system whose behavior never repeats itself and is always unpredictable and yet, paradoxically, always resembles itself and is infinitely recognizable.

1. Creating The Delayed Vector Space

I. DETERMINING THE DELAY TIME

System identification and prediction entails the intelligent and efficient way of deducing system parameters in order to effectively construct a dynamic model of a process. In simple terms it is much like knowing what goes in and out of a "black box", and then trying to re-construct the type of mechanism that is "working" in the box. The correlation between the observed and the modeled (forecasted and hypothetical) process is proposed to be used as the degree of determinability.

In some cases apparent randomness in time series may be due to chaotic behavior of a nonlinear deterministic system. In such cases it is possible to exploit the determinism to make short-term forecasts that are much more accurate than one could make from a linear stochastic model. This is first done by first reconstructing a state space and then using nonlinear function approximation methods to create a dynamical model. Nonlinear models are valuable not only as short term forecasters, but also as diagnostic tools for identifying and quantifying low-dimensional chaotic behavior.

To find how a system evolves from a given initial state one can employ the dynamic (equations of motion) incrementally along an orbit. The method of deducing the systems behavior requires computational effort proportional to the desired length of time to follow the orbit.

For example, systems such as a frictionless pendulum, the equations of motion may occasionally have a closed form solution, which is a formula that expresses

any future state in terms of the initial state . A closed form solution provides a short cut, a simpler algorithm that needs only the initial state and final time to predict the future *without* stepping through intermediate steps.

The unpredictable behavior of chaotic dynamical systems cannot be expressed in a closed form solution. Therefore there are no short cuts to analyze their behavior. Frictional losses for example cause the orbits to be attracted to a smaller region of the state space with a lower dimension. Any such region is called an attractor. An attractor is what the behavior of a system settles down to or is attracted to. A system may have several attractors. If that is the case, different initial conditions may evolve to different attractors.

In mathematical models of physical dynamical systems, the dynamical evolution is visualized in the state space whose dimension is given by the number of dependent variables. In experiments, the state space is usually not known beforehand and often only one variable of the system can be measured e.g. velocity. Thus only a projection of a trajectory of the system with usually high dimensional state space onto a single coordinate axis is given. It can be seen that the time series already contains most of the information about the total system and not just a minor part.

The single variable considered develops in time not due to its own isolated dynamical laws but is usually coupled to all other dependent variables of the system. Its dynamics, therefore, reflects the influence of all variables considered. This mutual interaction lets a single variable contain the dynamics of all the other ones. An example may help the reader. Take a simple oscillator: a mass on a

spring. The state space is the usual phase space given by the coordinates representing elongation x and the velocity, respectively . From our knowledge of the dynamics of the system, we know that with the knowledge of x(t) (one coordinate only) we also know information about the other state variables, since the state equations are coupled. Thus, when only one coordinate is measured, say the elongation x, then from x(t), the velocity (which is the other coordinate) can subsequently be determined [5].

The solution may be formulated as the problem of reconstruction of an attractor of a dynamical system from a time series of one measured variable only. In other words, if an attractor was, shall we say, three dimensional, being governed by three variables x, y and z and if x was the only measurable quantity, then we could say that impregnated within this quantity are the other two variables y and z.

Reconstruction of the state space of an attractor or nonlinear system from a time series of one (measured) variable only, can be solved using the concept of delayed vectors [5]. Theory states that for almost every state variable x(t) and for almost every time interval T, a trajectory in an m-dimensional state space can be constructed from the measured values of the time series; $[x(t_0 + kts), k=0,1,2,..,N]$, by grouping m values to form m-dimensional vectors. Therefore, in essence, the past behavior of the time series contains information about the present state. If for convenience, the sampling time τ is assumed to be uniform, this information can be represented as a *delay vector* of dimension m,

$$X_\tau(t_0+kts) = (x(kts),x(kts + \tau),.....,x(kts + (m - 1)\tau))^T$$

$$\text{for } k \geq 0$$

where t_S is the sampling interval at which samples of the variable x are taken and T denotes the transpose (by convention, all states are taken to be column vectors). The delayed vector space X_τ, is used in obtaining information about the future, using past information.

Delay vectors are currently the most widely used choice for state space reconstruction. Unfortunately to use them it is necessary to choose the delay parameter τ . Although Takens' theorem suggests that this choice is not important; however, because of noise contamination and estimation problems, it is crucial to choose a good value for τ . If τ is too small, each coordinate is almost the same, and the trajectories of the reconstructed space are squeezed along the identity line; if τ is too large, in the presence of chaos and noise the dynamics at one time become effectively causally disconnected from the dynamics at a later time, so that even simple geometric objects look extremely complicated. Examples of these behaviors are discussed later in this section.

Another method of state space reconstruction in common use is the principal components technique [16]. The simplest way to implement this procedure is to compute a covariance matrix $C_{ij} = (x(t - i\tau)x(t - j\tau))t$, where $| i-j | < m$, and then compute its eigenvalues, where $(\ )t$ denotes an average time. The eigenvectors of C_{ij} define a new coordinate system, which is a rotation of the original system. The eigenvalues of this matrix, are the average root mean square projection of the m- dimensional delay coordinate time series onto the eigenvectors. Ordering them according to size, the first eigenvector has the

maximum possible projection of the state vector, the second has the largest possible projection for any fixed state vector orthogonal to the first, and so on. This procedure can be useful in the presence of noise; discarding eigenvalues whose size is below the noise level can reduce noise.

Being the more popular of the two methods, the method of using delay coordinates was used. In the sections to follow are the techniques used to determine τ , the delay time and the effect of τ on the reconstruction of the state space.

As mentioned earlier the question of what is the best way to choose τ is still open. If τ is too small, then the coordinate at $x(n + \tau)$ and $x(n + 2\tau)$ represent almost the same information. Similarly, if τ is too large, then $x(n + \tau)$ and $x(n + 2\tau)$ represent distinct uncorrelated descriptions of the embedding space. Examples of this can be seen on pages 17 - 20 (Figures 6 - 9). Notice that as τ gets larger, the phase portrait becomes more evident. In other words, a larger amount of information is displayed. This will be true of any system; shown as examples are the Lorenz and Rossler Strange Attractors. The delayed phase portraits were obtained by plotting a pseudo-space portrait next to the original phase portrait. This was done using the Matlab program 'autocorr.m' (Appendix 9). The values of τ were arbitrarily chosen to demonstrate the effect of choosing the correct value of the delay time.

The Rossler and Lorenz differential equations (Appendix 5 & 6) were evaluated using the Runge-Kutta method, specifying the initial conditions, the start time, the time step, and the final time (Rk4.m; Appendix 4). This process provided the x,

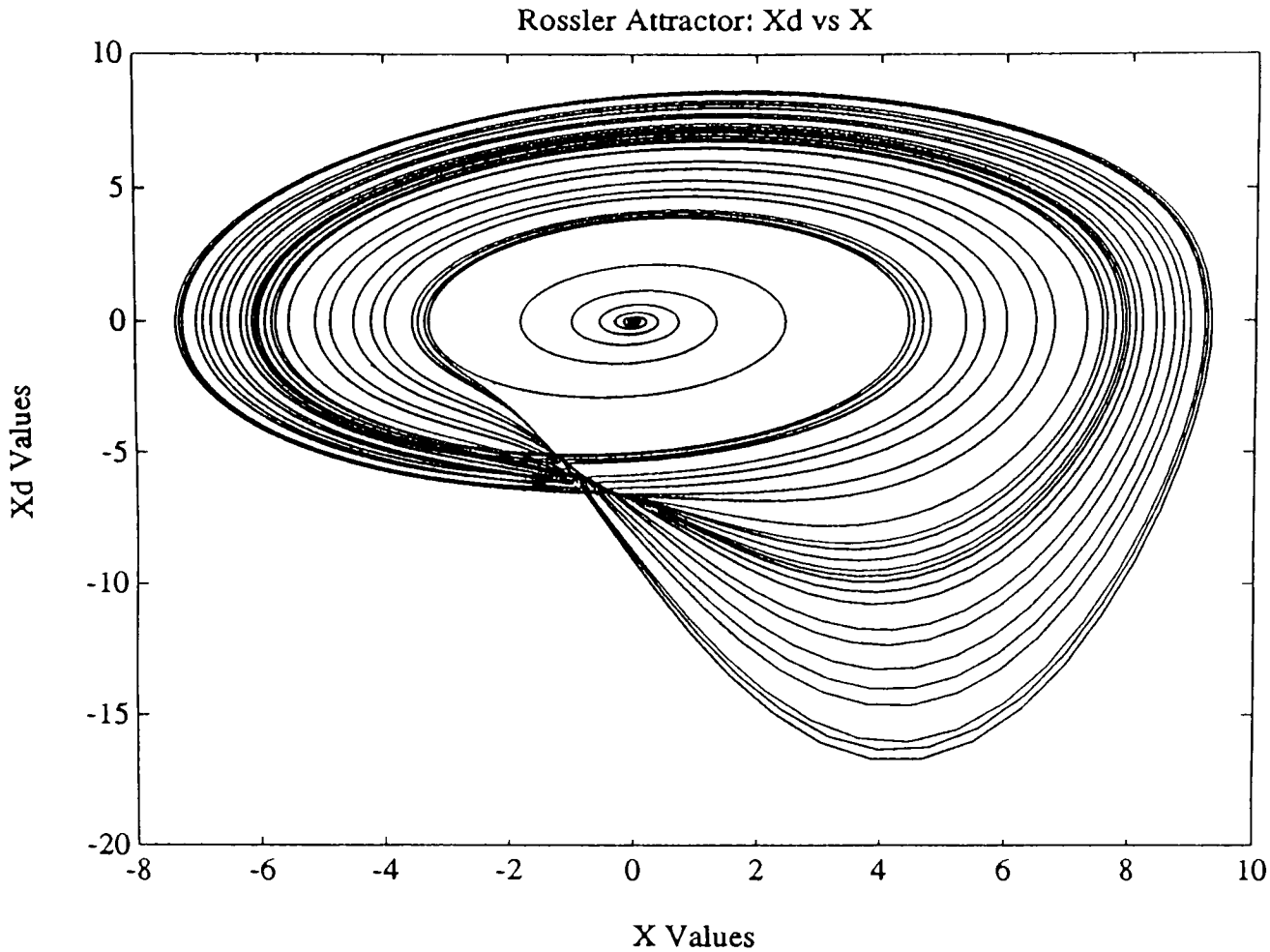


FIGURE 6

The original "Rossler Attractor". Shown here is the phase portrait of the 'x velocity' against the 'x' variable, derived from the original equations using a Runge-Kutta operation.

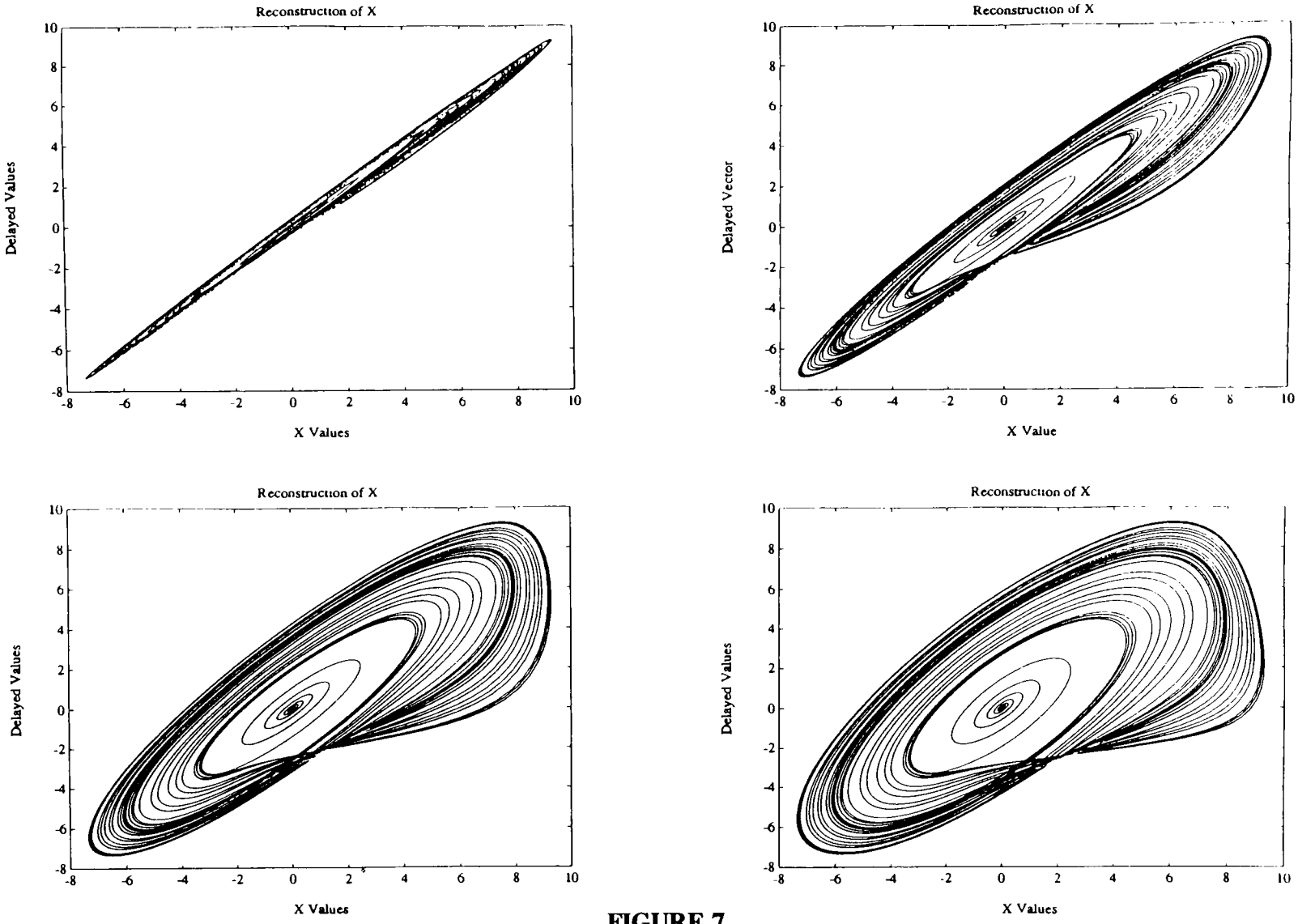


FIGURE 7

The Effects Of Using Different Delay Times: As the delay times become larger(from left to right) the delayed phase portraits evolve from a straight line to a projection of the original attractor. Here the delay times are $\tau = 2, 6, 8, 15$.

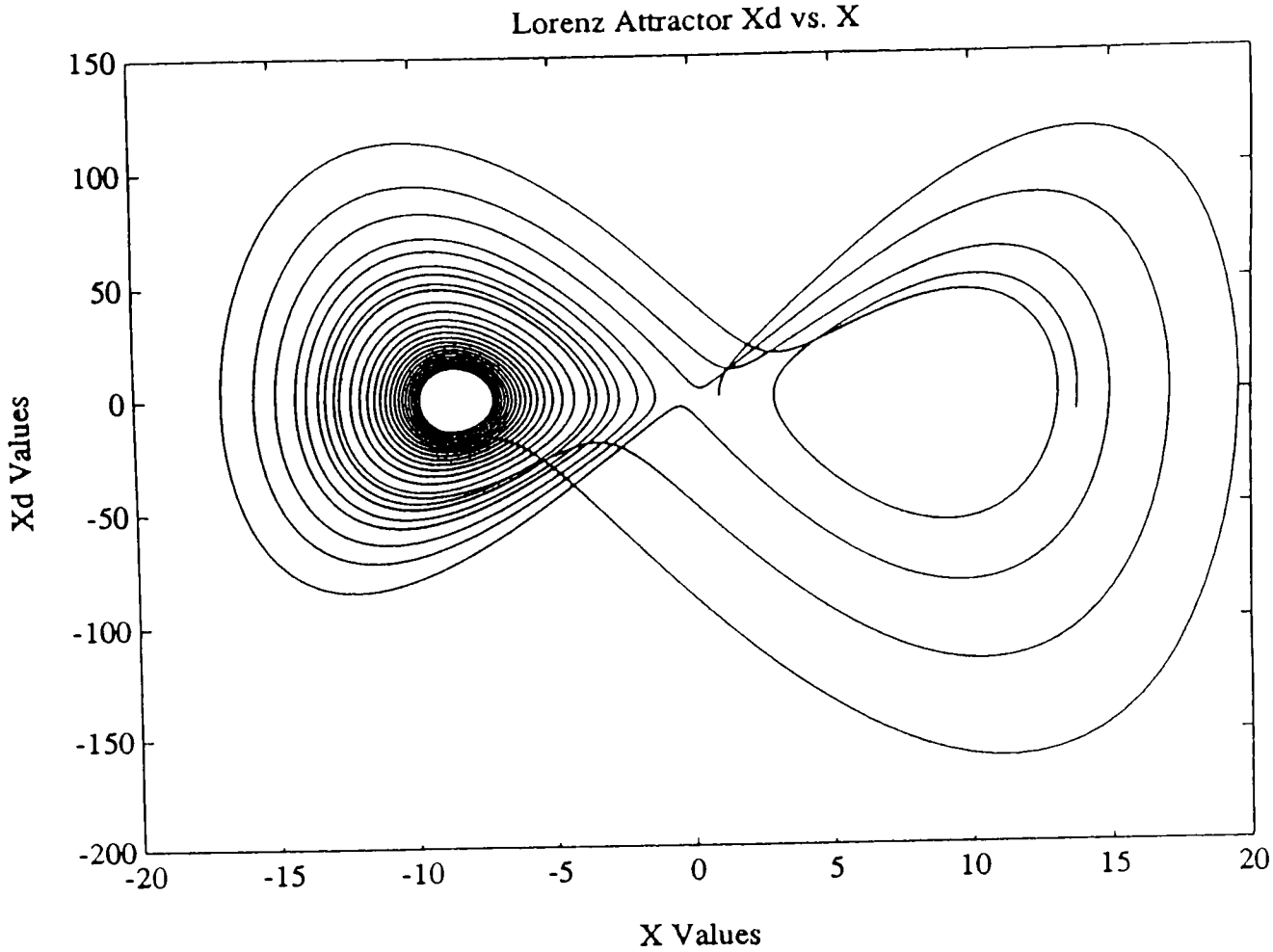


FIGURE 8

Part of the "Lorenz Attractor". Shown here is the phase portrait of the 'x velocity' against the 'x' variable, derived from the original equations using a Runge-Kutta operation.

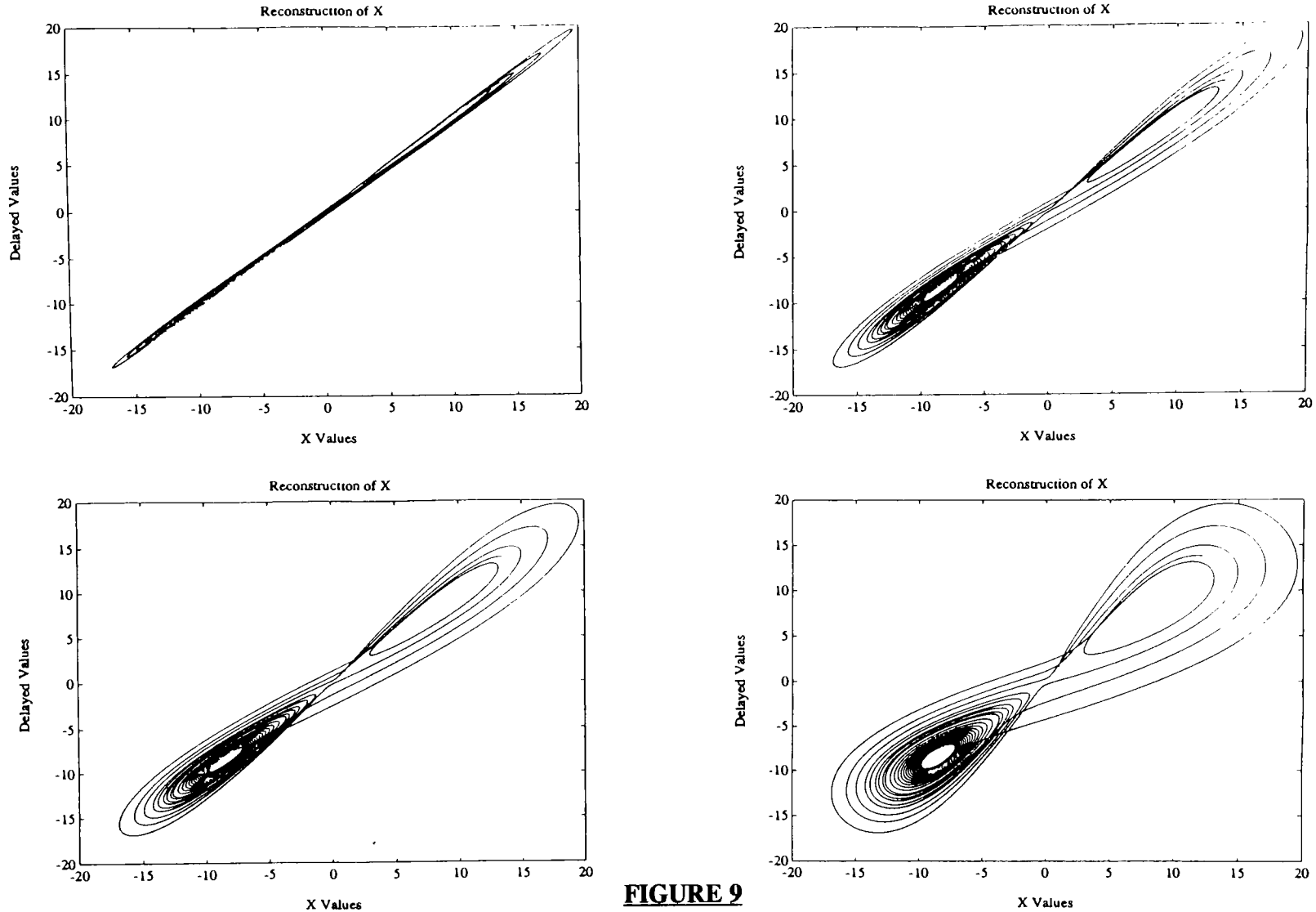
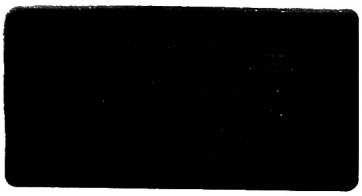
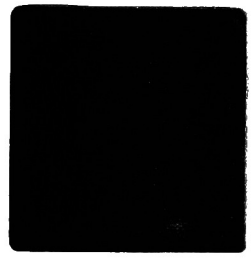
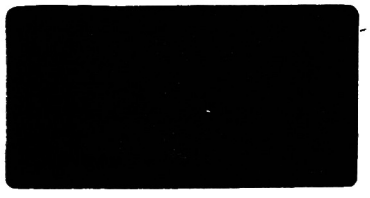


FIGURE 9

The Effects Of Using Different Delay Times: As the delay times become larger(from left to right) the delayed phase portraits evolve from a straight line to a projection of the original attractor. Here the delay times are τ = 2, 6, 8, 15.





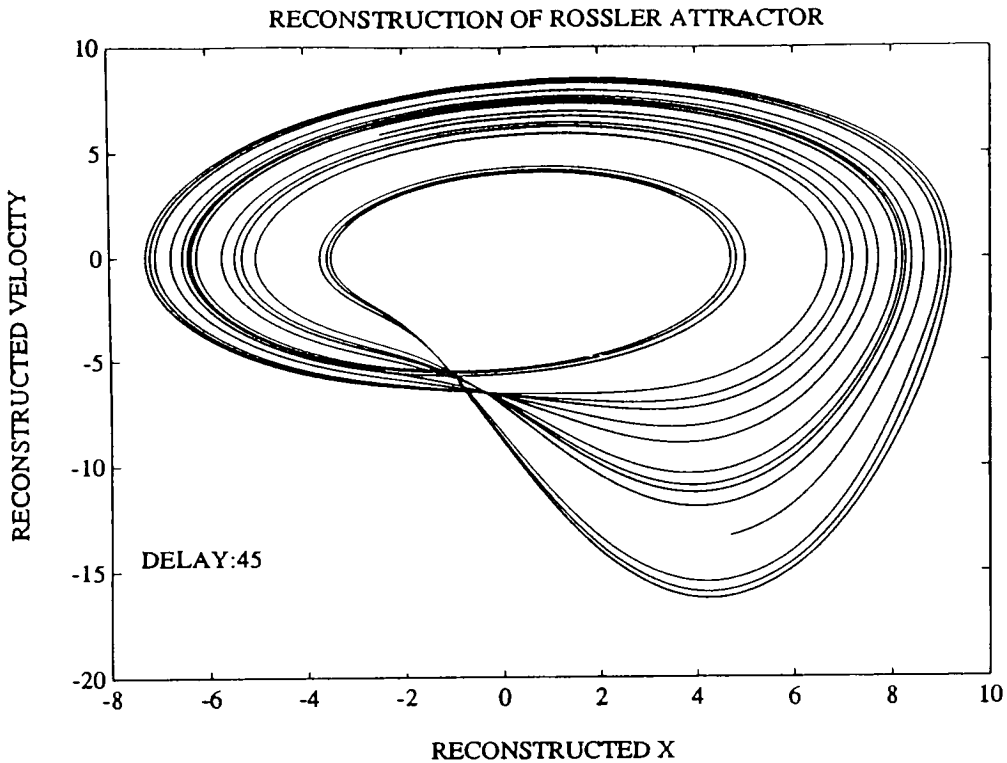


value would correspond to the delay time τ. The actual delay time used was usually a tenth of the value provide by this function [12] i.e the time associated with the first local minimum of the autocorrelation function. The value of this minimum point is usually a large value (too large to provide accurate information with regards to the system), therefore taking a tenth of this value establishes a time delay value that provides us with better results.

However, through the work that was done, it has been concluded that this is not always the best approach to follow. It was found by evaluating the decorrelation function in a higher dimension (usually the dimension of the system) that the optimum value of the delay usually fell in the region where the first maximum slope (or first inflection) of the autocorrelation function occurred.

The delay time provided by the first method was never sufficient for good curve fitting and usually resulted in bad curve fits. A better delay was always found to be much higher than was provided by this method. This however, does not say that the second method provided a value that corresponded to the optimal shift necessary to create the delay vectors. Using the second approach, we could confidently assume that the optimal delay time was in the near vicinity of this value. Examples to demonstrate these different delays(shifts) are shown on pages 23 - 26 (Figures 10 - 13), as are the effects each had on the reconstruction. As the figures show an improper choice of the delay, could lead to disastrous effects on the curve fitting.

The section that follows discusses the method used to determine, or rather estimate, the dimension of a nonlinear system using an obtained time series.



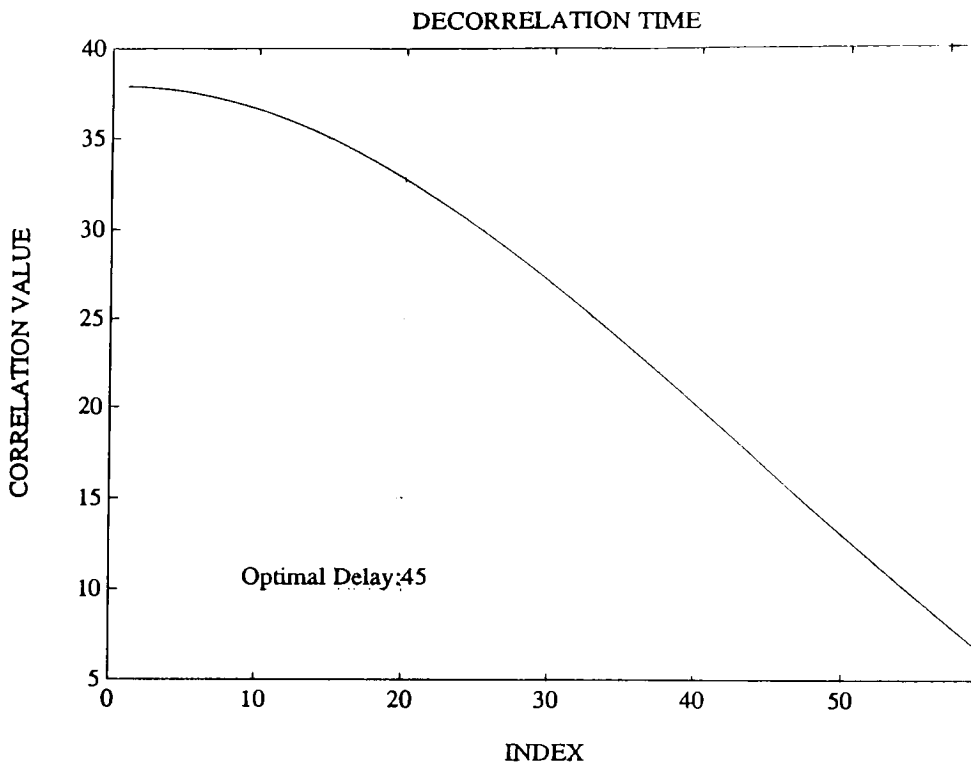
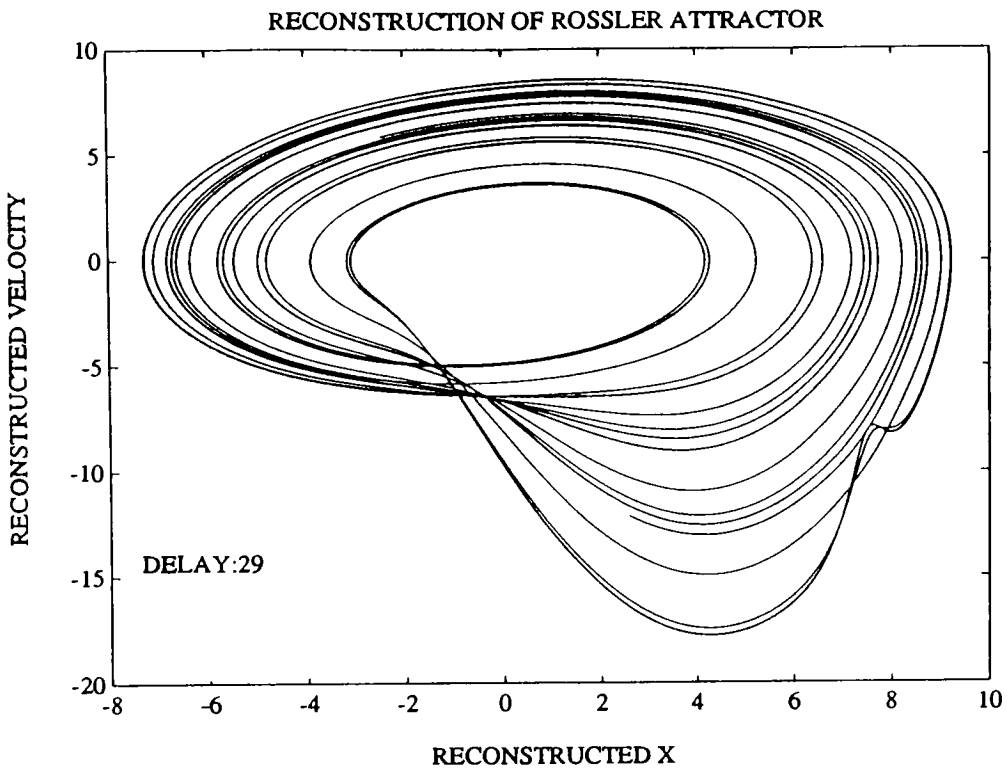


FIGURE 10

The effects of different shift values:

Shown here is a reconstruction of the Rossler attractor using the optimal shift value of 45. The value is obtained from the plot of the auto correlation. The maximum slope or first inflection point occurred at 45.



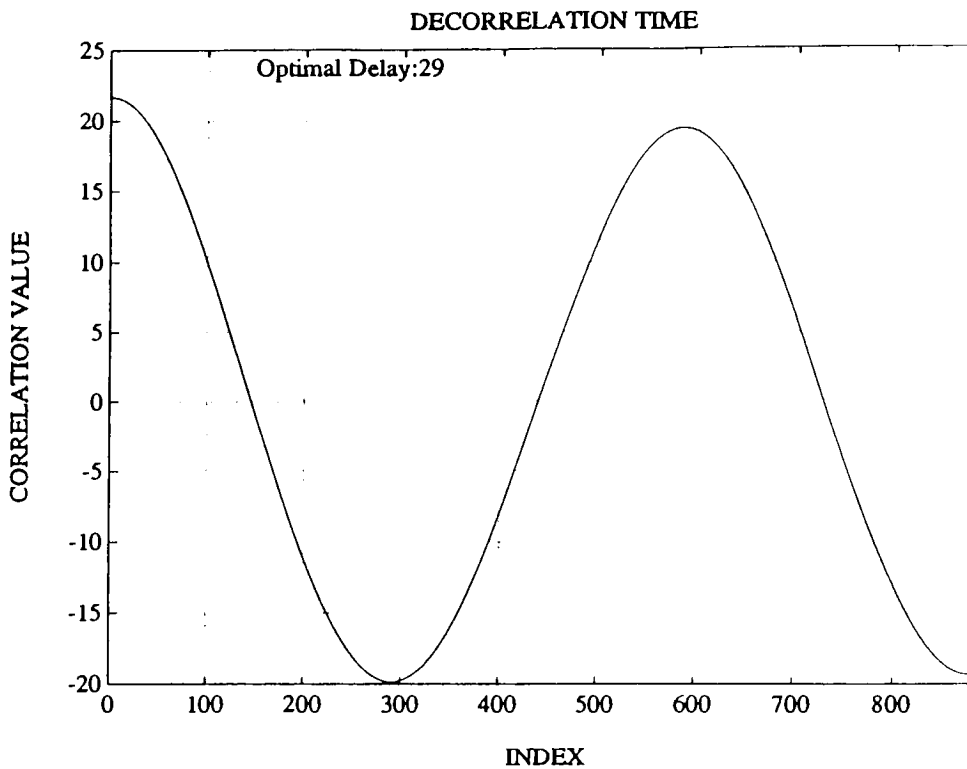
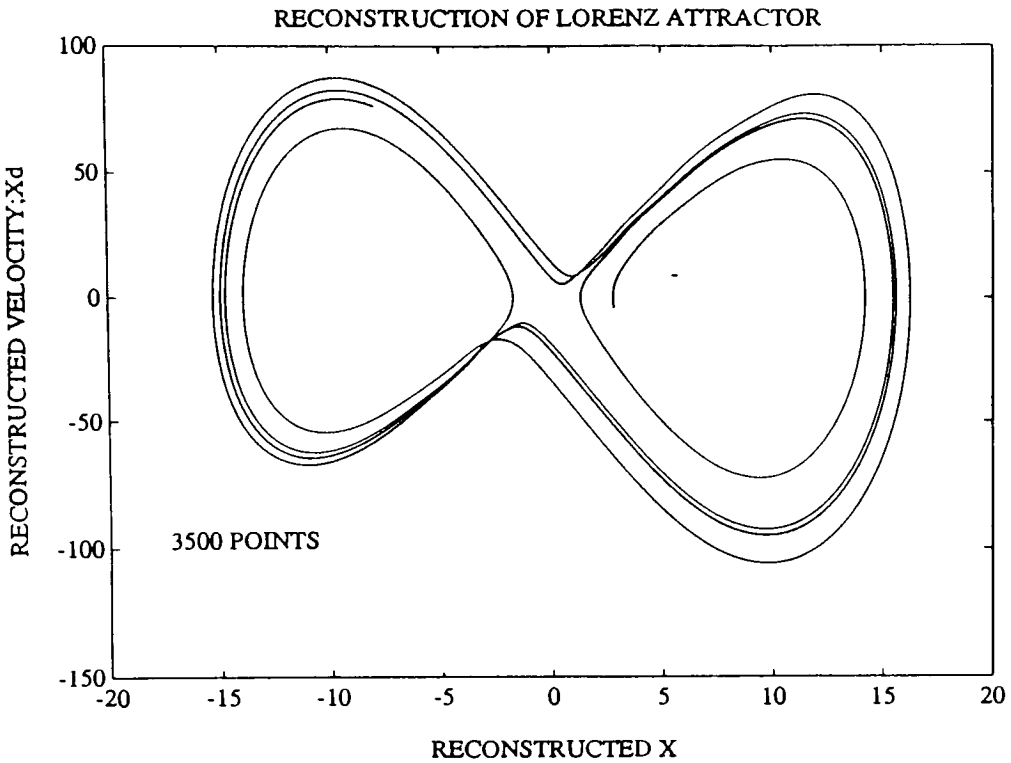


FIGURE 11

The effects of different shift values :

Shown here is another reconstruction, only this time using a tenth of the first minimum of the autocorrelation function (290). Note the glitch in the reconstruction.



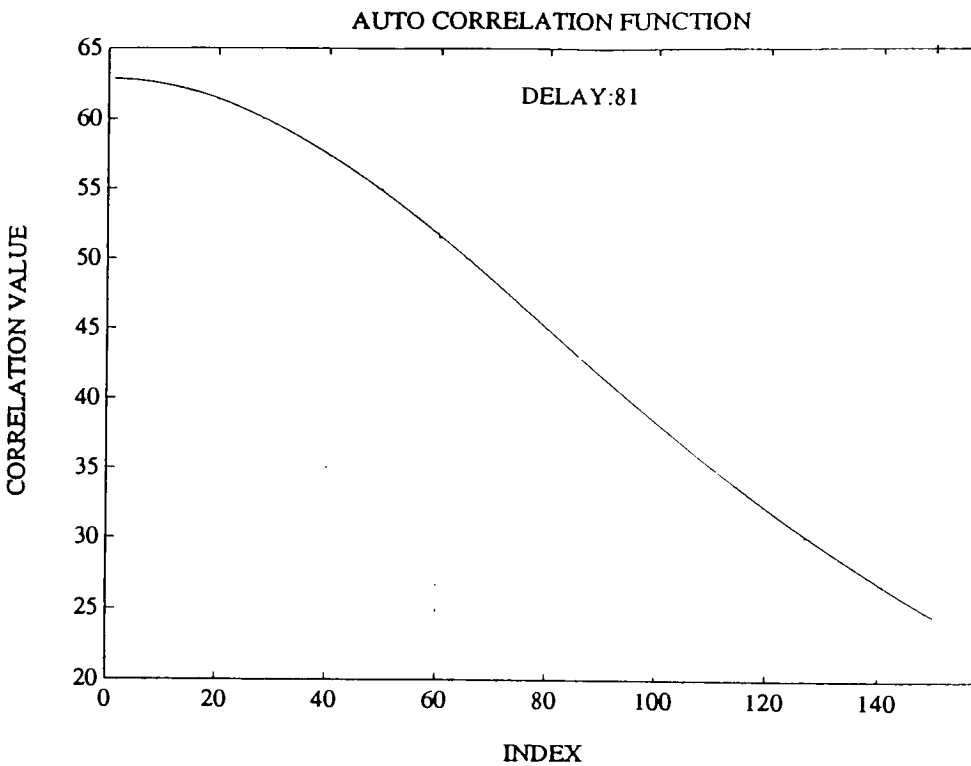
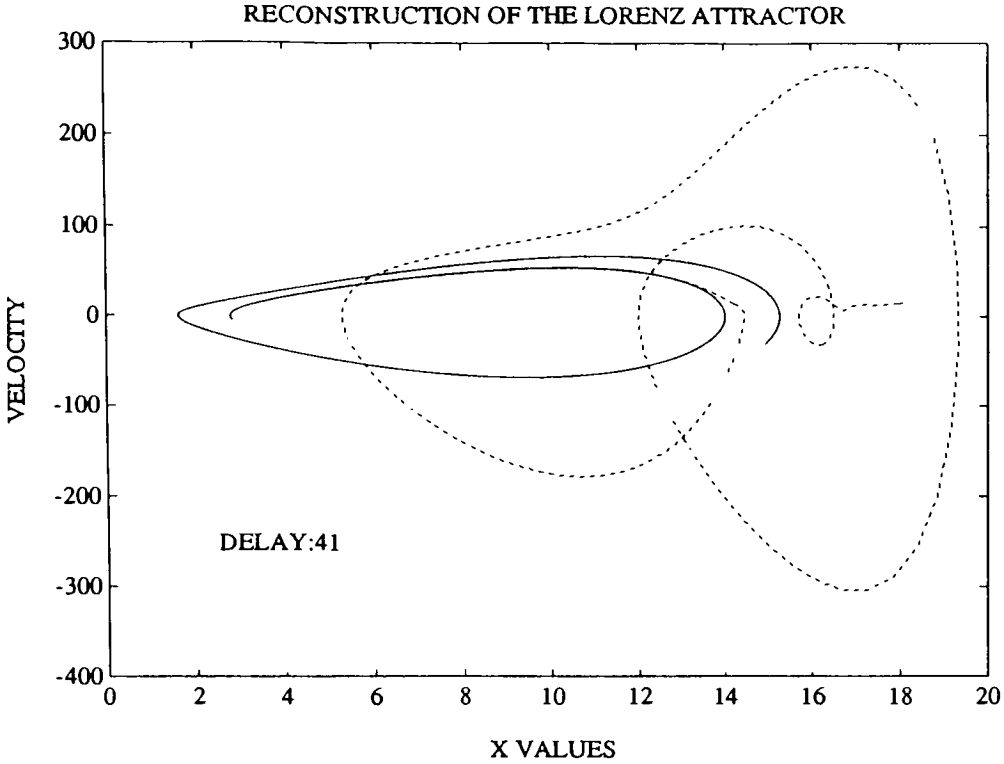


FIGURE 12

The effects of different shift values:

Shown here is a reconstruction of the Lorenz attractor using the optimal shift value of 81. The value is obtained from the plot of the auto correlation function. The maximum slope or first inflection point occurred at 81.



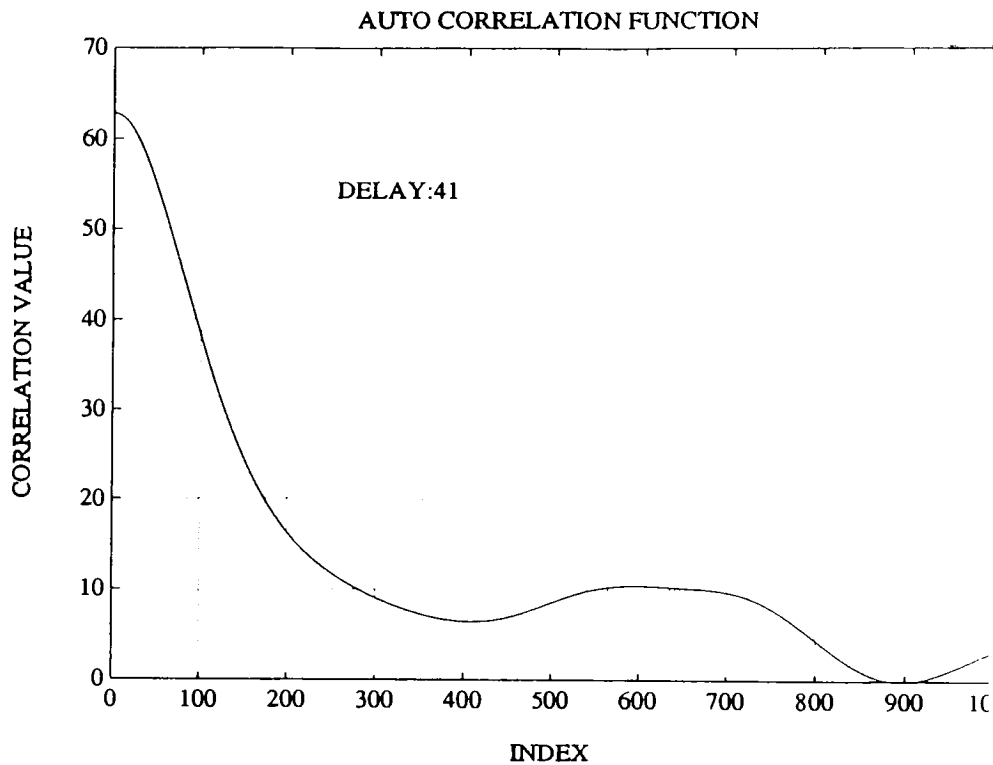


FIGURE 13

The effects of different shift values:

Using a tenth of the value where the first minimum occurs, the Lorenz attractor was reconstructed only to find that a very unstable solution was obtained. The reconstruction follows the original for about 500 points before it breaks away to infinty. The value used was 41 (a tenth of 410).

II. DETERMINING THE EMBEDDING DIMENSION

The development of algorithms for estimating the dimension of an attractor directly from a time series has been an active field of research over the last decade. The objective of these algorithms is to estimate the fractal dimension of a hypothesized strange attractor in the reconstructed state space.

The dimension of a system can be defined to be either:
- each pair of position-velocity coordinates associated with the displacement, or
- only one of the two elements, position or velocity.

Thus, a set of N bodies free to move in three spatial directions has 6N degrees of freedom. Sometimes it is easier to think of the phase space as containing all the degrees of freedom of a particular system.

It is known that complex aperiodic behavior can result from deterministic physical systems with few degrees of freedom. Dissipative dynamical systems which may have many degrees of freedom (such as fluids) can, after an initial transient time, settle down to a state in which only a few degrees of freedom are relevant to the dynamics. In this post-transient state, the system's trajectory through phase space is confined to a low-dimensional subset of the available phase space. When the subset is a strange attractor, motion is complex, aperiodic and typically chaotic. On the other hand, the motion of a dynamical system itself is complicated; a full description would require many degrees of freedom.

Thus an experimentalist, observing a system that displays apparent erratic motion, seeks to distinguish between these two kinds of motion, deterministic

"chaos" and stochastic "noise". Does the system have a (low dimensional) strange attractor in its phase space, and if it does, what is its dimension?

If the time series is deterministic and of finite dimension, the estimated dimension of the reconstructed attractor should converge to the dimension of the strange attractor as the embedding dimension is increased. If the time series is random, the estimated dimension should be equal to the embedding dimension. Unfortunately, these statements hold only in the limit of arbitrarily long noise-free stationary time series. In some cases, the convergence can be excruciatingly slow.

Historically, the first numerical algorithms were based on a "box-counting" principle, although this was found to be impractical for a number of reasons. The *correlation dimension* developed by Grassberger and Proccacia [9] and independently by Takens considers the statistics of distances between pairs of points, and this remains the most popular way to compute dimension. This method was used in our study to determine the dimension of a given system and will be discussed in greater depth as we proceed further on into this section.

Other approaches for estimating dimension include using the statistics of the k^{th} nearest neighbors, using Lyapunov exponents to estimate a "Lyapunov dimension", which is related to the actual geometric dimension, and using m-dimensional hyperplanes to confine the data in local regions [16]. It also has been suggested that prediction itself can provide a robust test for low-dimensionality. Basically, the smallest embedding dimension for which good predictions can be

made is a reliable upper bound on the number of degrees of freedom in the time series.

Going back to the Grassberger and Proccacia approach mentioned earlier, this method defines a *correlation integral* $C(r, N, d)$ that is an average of the pointwise mass functions $B(\mathbf{x}; r, N, d)$ at each point $\mathbf{x}$ in the reconstructed state space. Here, d is the dimension of the embedding space, N is the number of points, and $B(\mathbf{x}; r, N, r)$ is the fraction of points (not including $\mathbf{x}$ itself) within a distance r of the point $\mathbf{x}$. The asymptotic scaling of $C(r, N, d) \sim r^d$, for small r, defines the correlation dimension d. This point will be elaborated on further in the section.

The pointwise mass functions $B(\mathbf{x}; r, N, d)$ are kernel-density estimates for the natural measure of the strange attractor at points x_i on the attractor and as such more accurately characterize the attractor than the crude histograms that box counting provides. A further advantage is that the correlation integral scales as r^d down to distances on the order of the smallest distance between any pair of points; this range is significantly greater than box counting permits.

We wish to estimate the dimension of an attractor, which is embedded in an m-dimensional Euclidean space from a sample of N points on the attractor, that from the set $\{x_1, x_2, \ldots\ldots, x_N\}$ with x_i belonging to the embedding space $\mathbf{R}^m$. Grassberger and Proccacia suggest that we measure the distance between every pair of points and then compute the distance correlation integral:

$$C(r,N,d) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Theta(r - |x(i) - x(j)|)$$

In this expression , N refers to the number of points in the R^m embedding space of the attractor. represents the Heavyside step function given by:

$\Theta(x) = 0$ for $x<0$,

$\Theta(x) = 1$ for $x>0$,

$\Theta(0) = 0$.

The points of the attractor are represented by x_i and x_j, and the vertical bars $|\cdot|$ represent a suitable norm, in this case the magnitude of the "distance difference" between the i^{th} and j^{th} points in the delayed vector space i.e.

$$| x(i) - x(j) | = \sqrt{(x(i) - x(j))^2}$$

The dimension is established by counting the number of distances which are less than a fixed radius i.e.

$$N(R) = (\text{number of pairs with } | x_i - x_j | < R) \sim R^d$$

where d represents the correlation dimension of the attractor. The process is repeated as the radius is incremented from a minimum value to a maximum value. Thus r in the correlation integral represents the quantities mentioned above. At this point it is important to note as mentioned earlier, that for a sufficiently small r:

$$C(r) \sim r^d$$

Even though it not quite obvious at first sight , the statement referred to above shows us that 'd' can be determined from the slope of the curve obtained when C(r) is plotted against r , each on a logarithmic scale;

$$\log N = \log r^d,$$
$$\log N = d \log r,$$
$$d = \log N / \log r.$$

The quantity d, represents the minimum dimensionality of the artificial phase space necessary to include the attractor. If the time series consists only of the sampling of a single dynamical variable from a multivariable dynamical system, the correlation dimension still may be extracted from the series using embeddings consecutively in a sequence of higher dimensional spaces. Thus theory states that as we increase the embedding dimension, the slopes of these curves will eventually converge on the dimension of the attractor.

Finally if the attractor has a finite dimension, one can nevertheless construct the complete state vector x(t) by the following process:
$x(t + \tau)$ is used as the first coordinate, $x(t +2\tau)$ as the second and $x(t + n\tau)$ as the last, then if n is large this will be a faithful representation of x initial. Different choices of n and of the delay time τ will lead to differently shaped attractors, but all of them will have the same dimension.

The Matlab program "dimembed.m" (Appendix 11) was written to evaluate the dimensions of the Rossler and Lorenz attractors. There was a minor difference

however; each time the program embedded in a higher dimension, the optimal delay time was computed for that particular embedding dimension. This was done so that each time the program ran, the most amount of information was extracted from the time series for a better approximation of the dimensionality of the system in question.

The embedding program provided us with the following results; using a data set of 15,000 points (time step 0.002s) the dimensionality of the Rossler attractor turned out to be approximately 1.98. The expected value is slightly greater than 2.0. The Lorenz attractor was also run with 15000 points (time step 0.002s) and in this case the dimensionality was found to be 2.03. This is compared with the accepted value of 2.05.

The Grassberger and Proccacia method of determining dimensionality is only good for systems with low dimensionality. Furthermore, the process is painstakingly slow. To get good comparable results, it is required that approximately 20,000 points or more be used in the process, as the greater the number of points there are, the more information there is about the system. This does not help at all in the run time of the program, and as would be expected, increases it tremendously.

Once the dimensionality has been determined, we are ready to develop the model equations that will essentially capture the essence of the system in question. The calculated dimensionalities are rounded off to the next highest integer (i.e. a system with a dimensionality of 2.05 would be best described using a three dimensional system).

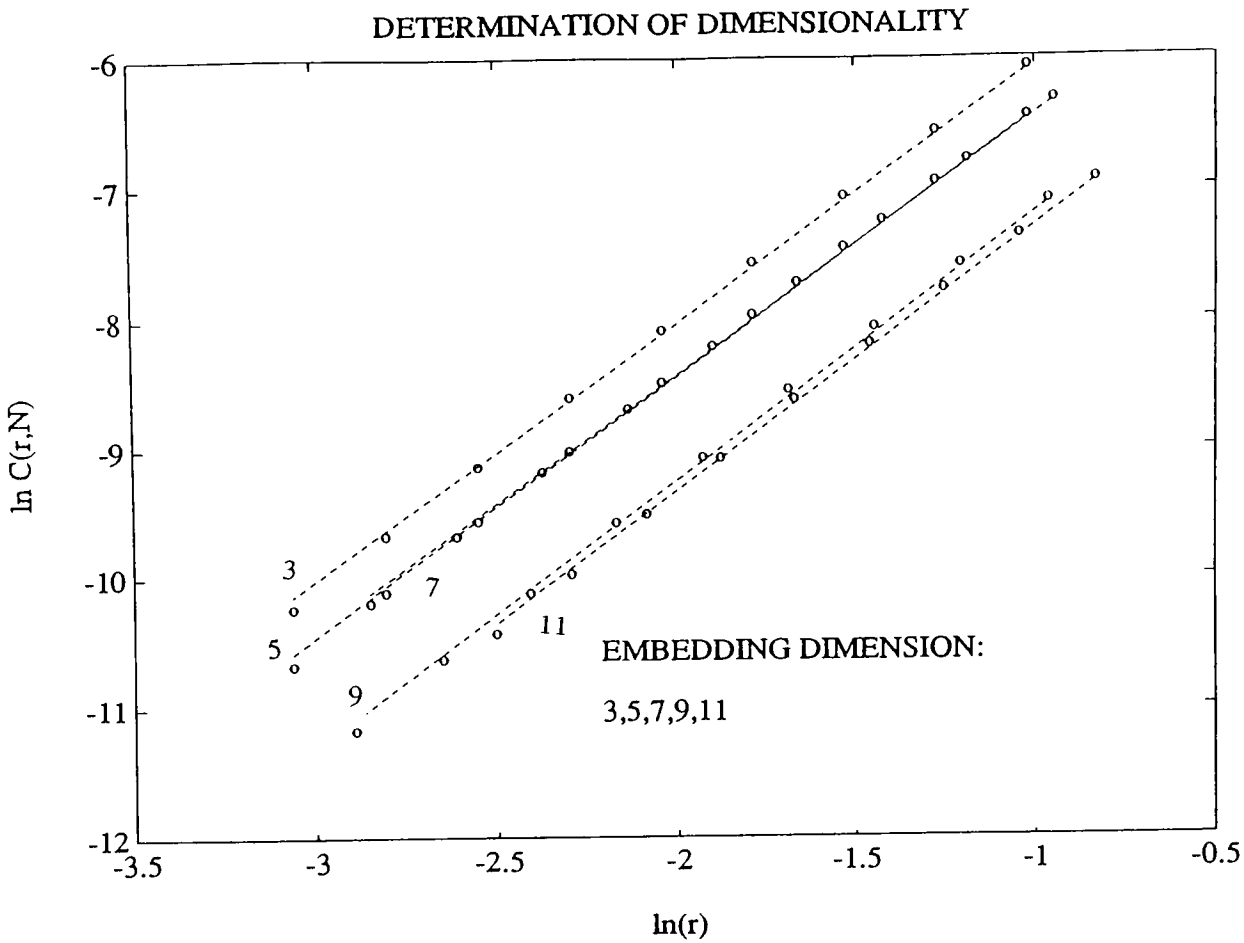


FIGURE 14

Determining the dimensionality of the Rossler attractor: the embedding dimension was carried out up to 11, because it is a time consuming process. In any case, note how the lines are beginning to show a trend. The slope of the last line (embedding dimension 11) is approximately 1.98.

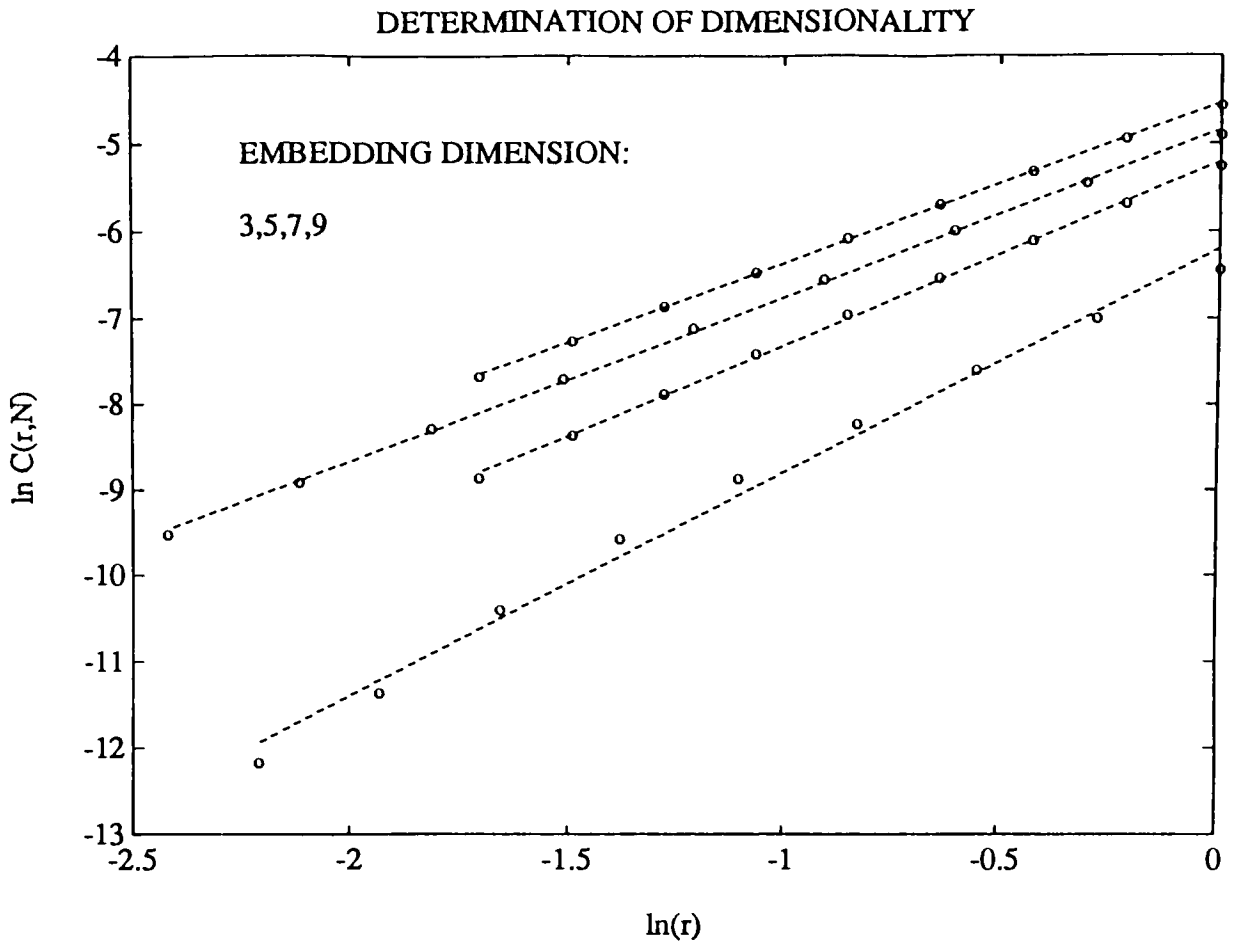


FIGURE 15

Determining the dimensionality of the Lorenz attractor; this also was only carried out to 9 dimensions (taking approximately 10 Hrs to run 15000 points). The lines began to converge on the the dimension of the system. The value of the slope of the last line is approximately 2.03.

2. Deriving Nonlinear Equations To Approximate The System

EXTRACTION OF DYNAMICAL EQUATIONS FROM DATA

It has generally been assumed that quantities that fluctuate with time or space with no discernible pattern could be described by a large number of deterministic equations or by stochastic ones. Ideally, one would like to be able to extract equations from a fluctuating time series. But due to the lack of additional information, this becomes quite unrealistic. The variable observed may not be simply related to the fundamental dynamical variables of the system. The measurement will usually be contaminated by noise and round-off errors and limited by sample rate and duration.

However, it may be possible to find a system of equations which mimic the general features such as the topology in a suitable phase space. These equations might shed insight into the behavior of the system.

The goal of forecasting is to predict the future behavior of a time series from a set of known past values. Once we have chosen a method for state space reconstruction, the data set can be mapped into state space. We can construct a model for the dynamics by using an approximating function F.The approximation F also can be used to predict future states.

The use of a state space representation is most effective when the dynamics is stationary or can be mapped onto something that is stationary. The phrase "stationary", refers to an autonomous system, that is no time-varying coefficients. Forecasting involves *extrapolation* in time, in the sense that one uses data from one domain (the past) to extrapolate to the behavior of the data in a disjoint domain (the future). Extrapolation is inherently a difficult problem. But as we

have seen through earlier discussions, state space reconstruction makes it possible to convert extrapolation of the time series into a problem of interpolation in the state space.

The most commonly used approach of time series forecasting assumes that the dynamics F is linear. This has several advantages. Linear functions have a unique representation, and they are relatively easy to approximate. However, as is known, linear systems are exceptional; virtually any real system contains nonlinearities. Furthermore, linear dynamical systems cannot produce limit cycles or chaotic dynamics. Thus to model chaotic dynamics, we are forced to consider nonlinear functions.

The importance of nonlinear functions cannot be overemphasized. More recently, it has been appreciated that ordinary, but nonlinear, differential equations with as few as three degrees of freedom or difference equations with a single degree of freedom can have pseudo-random (chaotic) solutions. This has led us to the hope that such simple systems can model the real world.

We will now take this opportunity to discuss a few aspects that we have found useful, first discussing several nonlinear representations and then discussing the methods we used to characterize the chaotic systems that were studied.

Any topologically complete set of special functions can be used to represent nonlinear functions. Discussed below are some of the various techniques that can be employed in function approximation problems.

o *Wavelets* are a localized generalization of the Fourier series [16]. Their most immediate use is as a means of signal decomposition and, hence, state space reconstruction. They also present an interesting possibility as a function representation within the state space.

o *Neural nets* are currently very popular [16]. There are many different varieties of neural nets, and it is beyond the scope of this investigation to discuss them. Neural nets have the disadvantage, when compared to some other methods of function approximation, that for most algorithms parameter fitting is slow.

o *Radial basis functions* are of the form:

$$f^*(s) = \Sigma\, a_i \, \emptyset \, (\| s - s_i \|)$$

where $\emptyset$ is an arbitrary function, s_i is the value of s associated with the i^{th} data point, and $\| s - s_i \|$ is the distance from s to the i^{th} data point [16]. This functional form has the advantage that the least squares solution of a_i is a linear problem.

o *Local Function Approximation* is carried out by fitting parameters based on points within a given region [14]. This approach allows for considerable flexibility in building a globally nonlinear model while fitting only a few parameters in each local patch. Local fitting has the advantage of being quick and accurate, but has the disadvantage that the approximations are discontinuous.

o *Polynomials* are of the form

$$f^*(s) = \sum a_1 \bullet s_1^{P1} \ldots\ldots\ldots a_d \bullet s_d^{Pd}$$

(where d represents the dimension of the variable s).

They have the advantage that the parameters 'a' can be determined by a least squares algorithm, which is fast and gives a unique solution [16]. They have the disadvantage, in that the solution usually approaches infinity, as s --> ∞ and consequently do not extrapolate well outside the domain of the data set. Furthermore , they behave poorly under iteration.

For our purposes, we decided to use polynomials for fitting the coefficients to the data set. The choice was made due to the ease of its implementation. This provided us with a unique dynamical equation that mimicked the original data set to approximately 1000 points.

We shall now outline the method used for determining a set of differential dynamical equations of motion which approximate an underlying generating process. Our method can recover a system of equations with good predictive ability using a few data points.

Given a (scalar) time series x [t] derived from some physical system as a signal, we assume that at least one component of the signal has been generated by physical process containing a few nonlinear degrees of freedom and is chaotic, although this not an absolute requirement. Using the time-delay phase space reconstruction methods described in the first section, we construct a Euclidean space that is topologically equivalent to the original system phase space by forming the delay vector space (as discussed earlier). To do this, we need to know the embedding dimension 'd' and the delay time τ. The methods used to determine these parameters have already been discussed in the previous sections.

We assume that motion in the original phase space is governed by a set of coupled ordinary differential equations (ODE). We attempted to find an approximation to this set of ODE's in the reconstruction phase space(let us call this delayed vector space, X_τ) capturing the underlying dynamics in a simple set of global equations, which are valid globally on the data attractor and which have the form:

$$dX_\tau[t]/dt = F(X_\tau[t]) \qquad \text{or}$$

$$\dot{X}_\tau = F(X_\tau[t]).$$

Calculating the derivatives of X_τ was performed using a sixth order accuracy central differencing technique. It was implemented using the Matlab code, 'cent6.m' (Appendix 1 & 12).

The next step was to determine the ODE. We assume a basis set of the components of a vector argument, the simplest case being polynomials, with:

$$F(X_\tau[\,t\,]) = a_{i1}p_1(X_\tau[\,t\,]) + a_{i2}p_2(X_\tau[\,t\,]) + \ldots\ldots + a_{iM}p_M(X_\tau[\,t\,]).$$

Here p_j represents the terms of an O^{th} order, m-dimensional polynomial constructed from the elements of $X_\tau[\,t\,]$ and the a_{im} are the coefficients of these terms. The terms generated using the programs, were of the fifth order. In all, there were approximately 56 terms.

Using the Matlab program 'nlfit.m' (Appendix 13), the required polynomial is generated. The basic core of the program generates a matrix which represents all possible combinations of the $X_\tau[t]$ values required to provide an adequate nonlinear fit to $\dot{x}(t)$, the approximated values of $\dot{x}(t)$.The program works such that all power combinations maybe generated using a simple nested algorithm. As the program cycles through each loop new unique polynomial terms are generated.

For basis sets such as simple polynomials that are linear in the coefficients, the system of equations can be written as a set of matrix equations,

$$A_i\, X_\tau = \dot{x}_i \ .$$

In the above representation, the k^{th} row of X_τ consists of the p_m generated by the vector $X_\tau[t]$ as described above, A_i is the matrix of the unknown coefficients for the function F, and $\dot{x}_i$ is a column matrix of approximations to the local derivatives in the i^{th} direction of the phase space trajectory at each data point $X_\tau[t]$. Typically to obtain A_i (and therefore F), we invert this matrix equation using a least squares minimization method such as the singular value decomposition (SVD).

The program has been written to generate 56 different terms. As mentioned earlier using the original Rossler or Lorenz equations we were able to generate original values of x, y, and z based on specified initial conditions and a Runge-Kutta routine. Thus in the same manner to check how good the generated ODE is, we subject the equation or rather the coefficients to a Runge-Kutta routine to iterate and generate new predicted values (z[t]) for each system. Required for the iteration are the values for the initial conditions, the time step and the number of points required to be generated. The Matlab program 'nlpoly.m' (Appendix 14) was written to perform this procedure within a Runge-Kutta routine (Appendix 15).

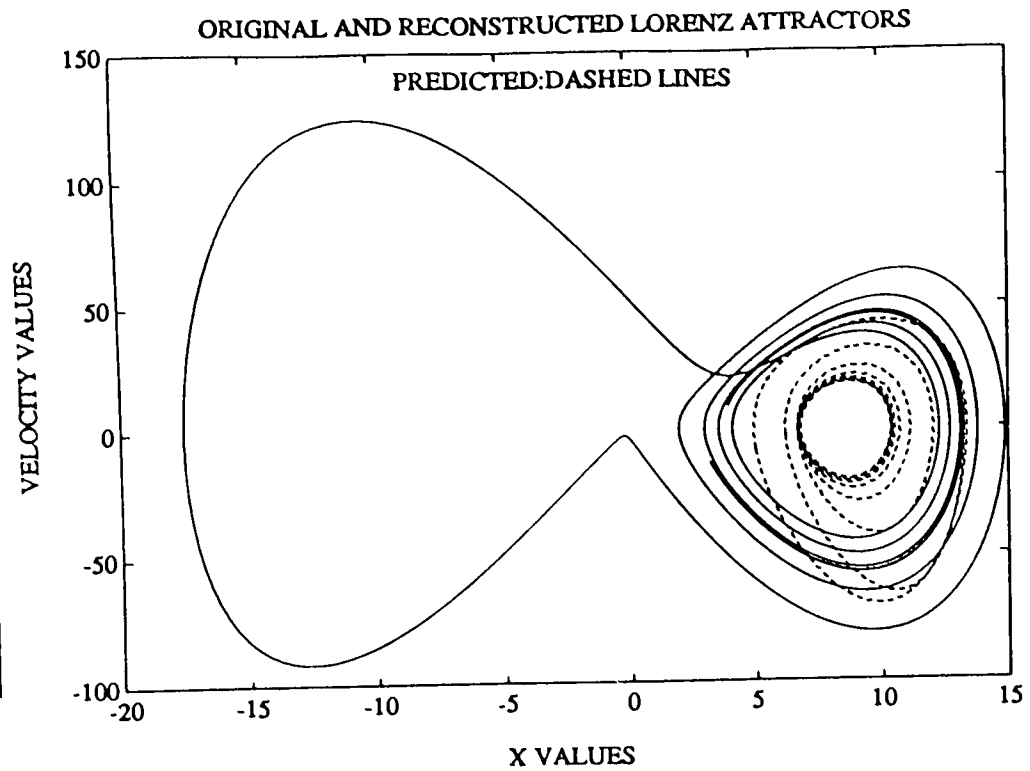
Since phase portraits are plots of the velocity values () versus the x[t] values, therefore if it is required to compare attractors, new values of corresponding to the new predicted values, z[t], need to be generated. Using the program 'nlpvel.m' (Appendix 16), these new derivative values are determined. Different polynomial combinations of z[t] are constructed, as was carried out in the program 'nlfit.m'. Thus using the coefficients determined earlier for the least squares fit to the derivative values ('cent6.m'), we are able to calculate new $\dot{x}$ values. That is,

$$\dot{x}_{new} = A_i \cdot z.$$

Using the program 'recon.m' (Appendix 17), we encapsulated all the steps required to reconstruct the Rossler and Lorenz systems. Briefly summarizing; we begin by constructing the delayed vector space. Using the central difference approach we obtain a nonlinear function that describes the derivative values of the delayed space. Then using Runge-Kutta we work backwards to obtain the new reconstructed x values. If we wish to compare phase portraits, we then use the unique coefficient matrix and these x values to determine the new reconstructed velocities.

The next few figures 16 - 22, demonstrate the results obtained from using this method of reconstruction. The systems shown are the Lorenz and Rossler systems and their reconstructed models. For both systems we were able to obtain models that followed the original system for approximately 1500 points before the error begins to visibly accumulate. However as Fig. 16 would indicate the reconstruction can sometimes be quite disastrous.

Once the ODE has been obtained, it is now necessary to study exactly how long they can be used to predict the future to reasonable accuracy.



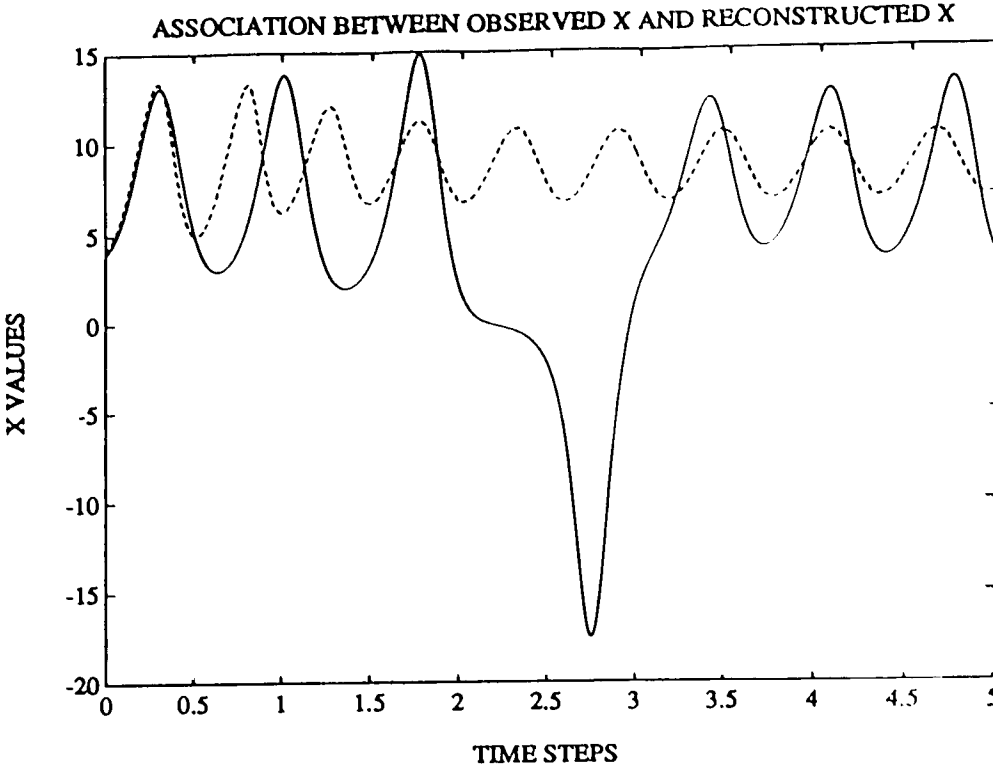
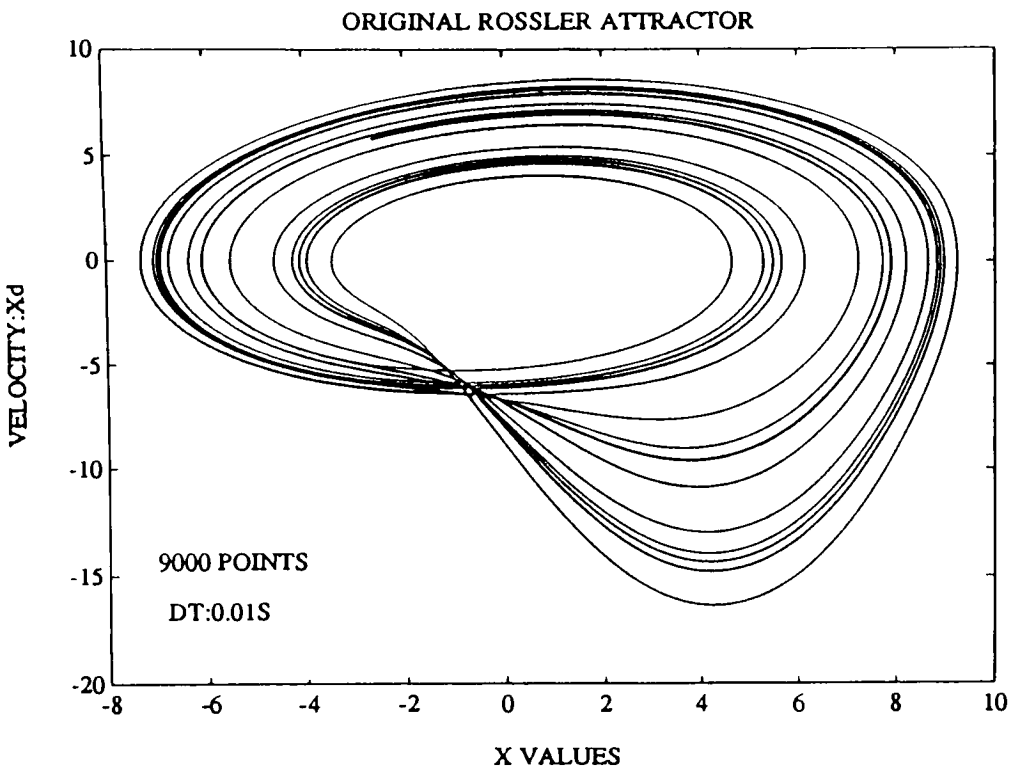


FIGURE 16

This figure illustrates the importance of a good delay required for the reconstruction; in this case a poor reconstruction of the Lorenz attractor. The time delay used was too small (delay steps: 36)and thus there was not adequate information for a good reconstruction, thus the collapse to the basin of the attractor. The figure also shows the association between the reconstructed and observed values of x; the predicted being represented by the dashed lines.



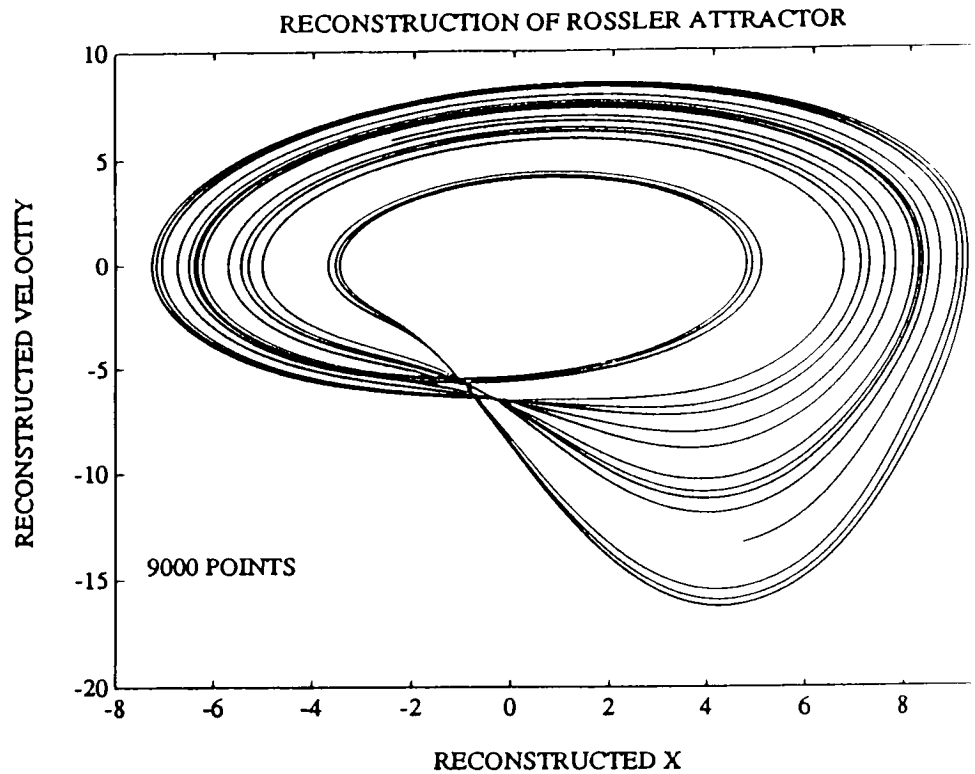


FIGURE 17

Comparison between the original attractor and the reconstructed attractor:

The optimal time delay used in this reconstruction was 45. Note that it appears that the reconstruction is not as complex as the original, this is because of the loss of information when reconstructing using delayed vectors.

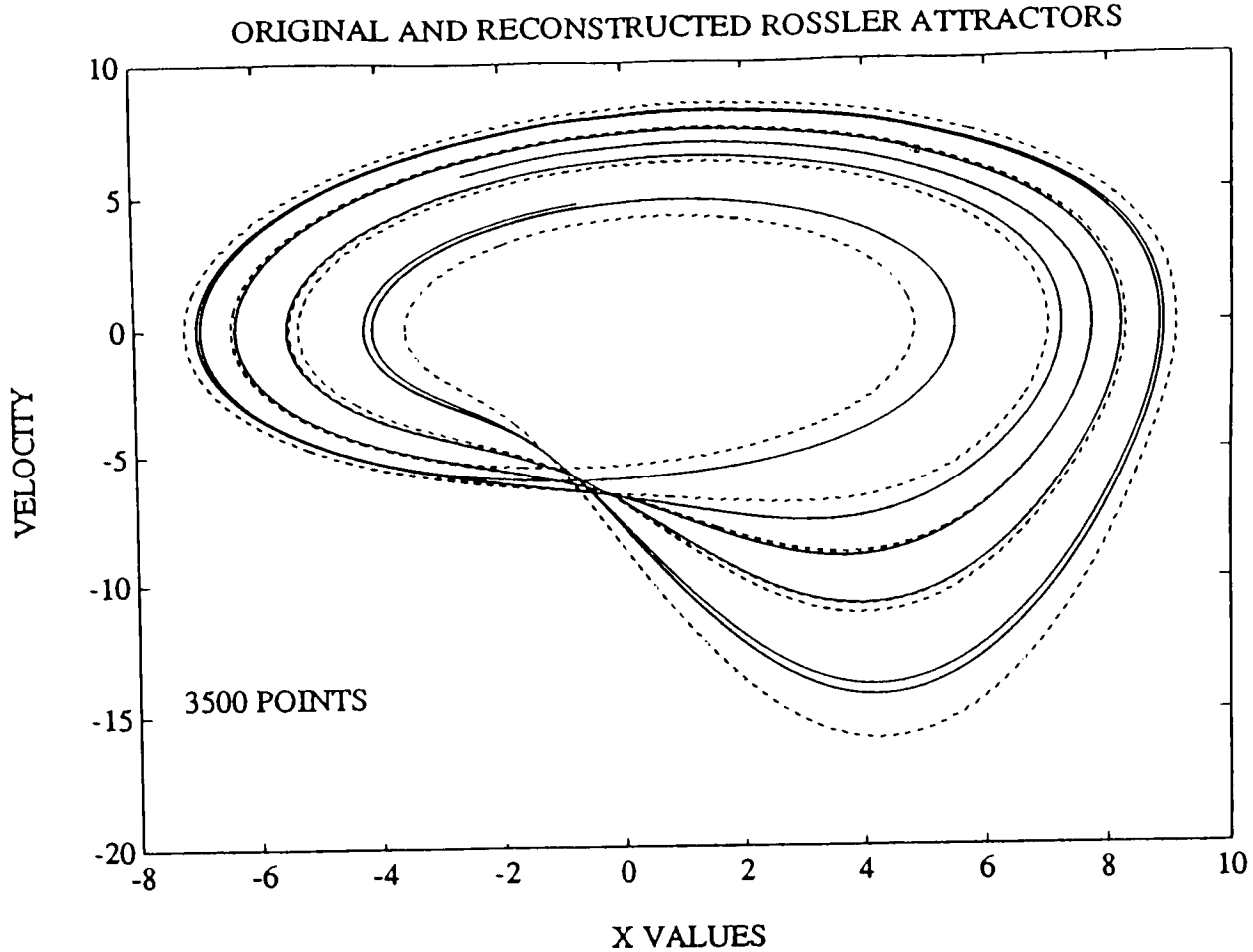
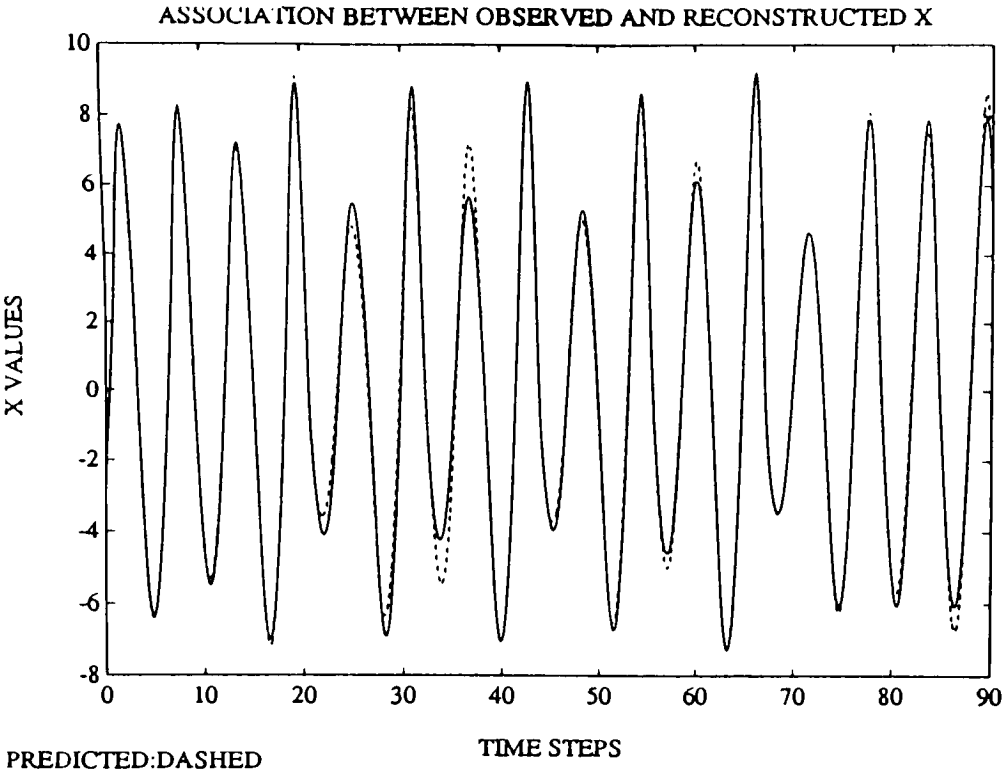


FIGURE 18

This figure shows how good the reconstruction actually is; only 2500 reconstructed points have been shown to reduce the complexity of looking at both original and the reconstruction on the same figure. The reconstructed attractor follows the original for approximately 1500 points before it breaks away.



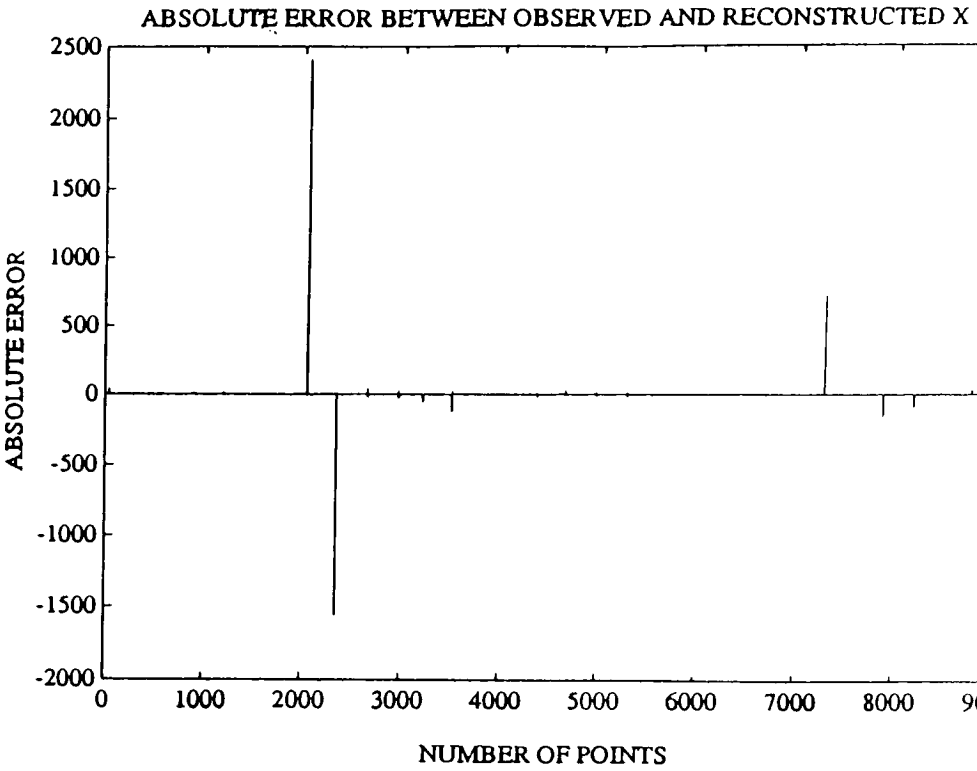
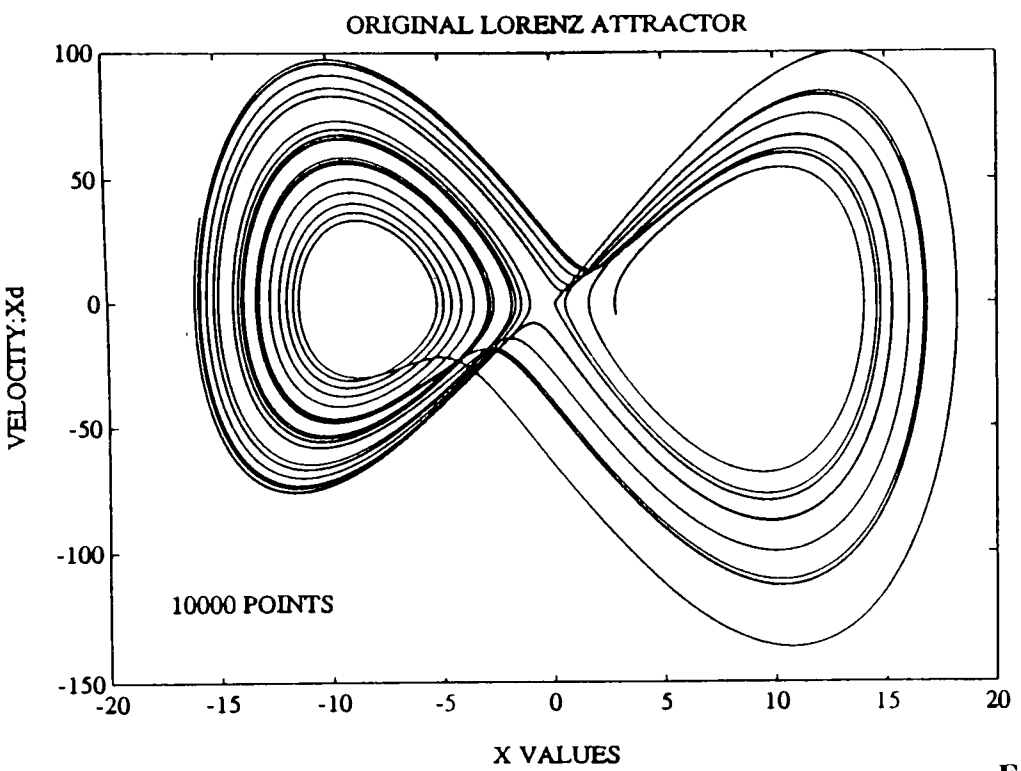


FIGURE 19

(Left to Right) Here we visually get a feel for the difference in the values between the actual x values and the reconstructed x values used to create and recreate the phase plots. As can be seen the values are extremely close for the first 1500 points. Also shown is a plot of the absolute error associated with these two variables.



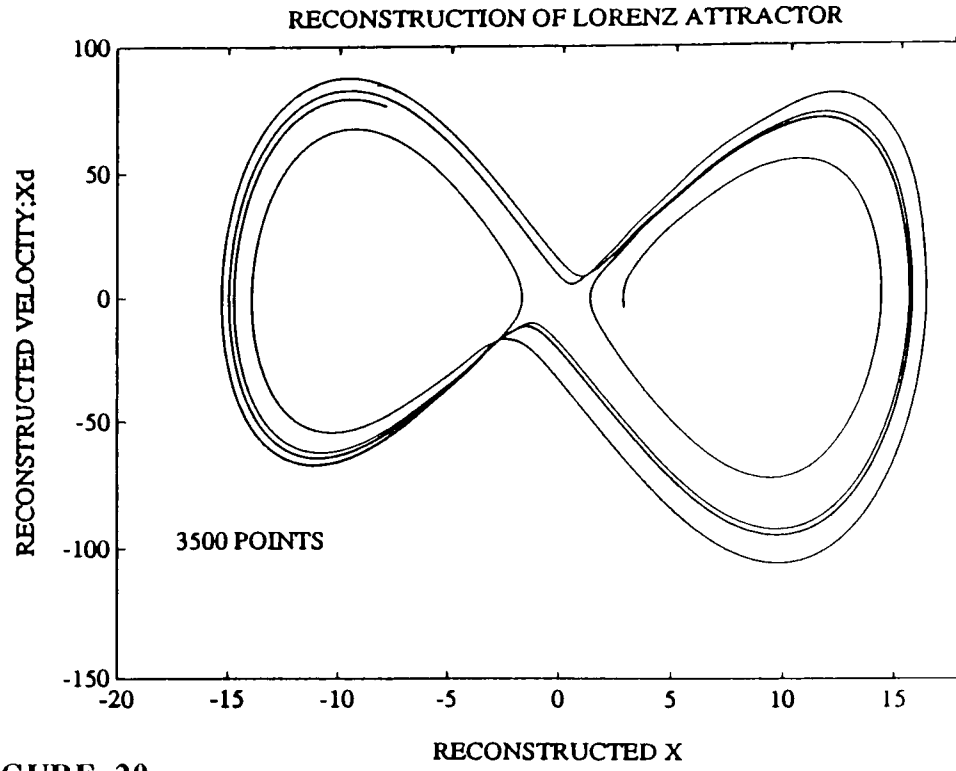


FIGURE 20

Comparison between the original attractor and the reconstructed attractor:

The optimal time delay used in this reconstruction was 81. The Lorenz attractor is much harder to reconstruct. For good reconstruction, in this case it was necessary to use a small time step of 0.002s.

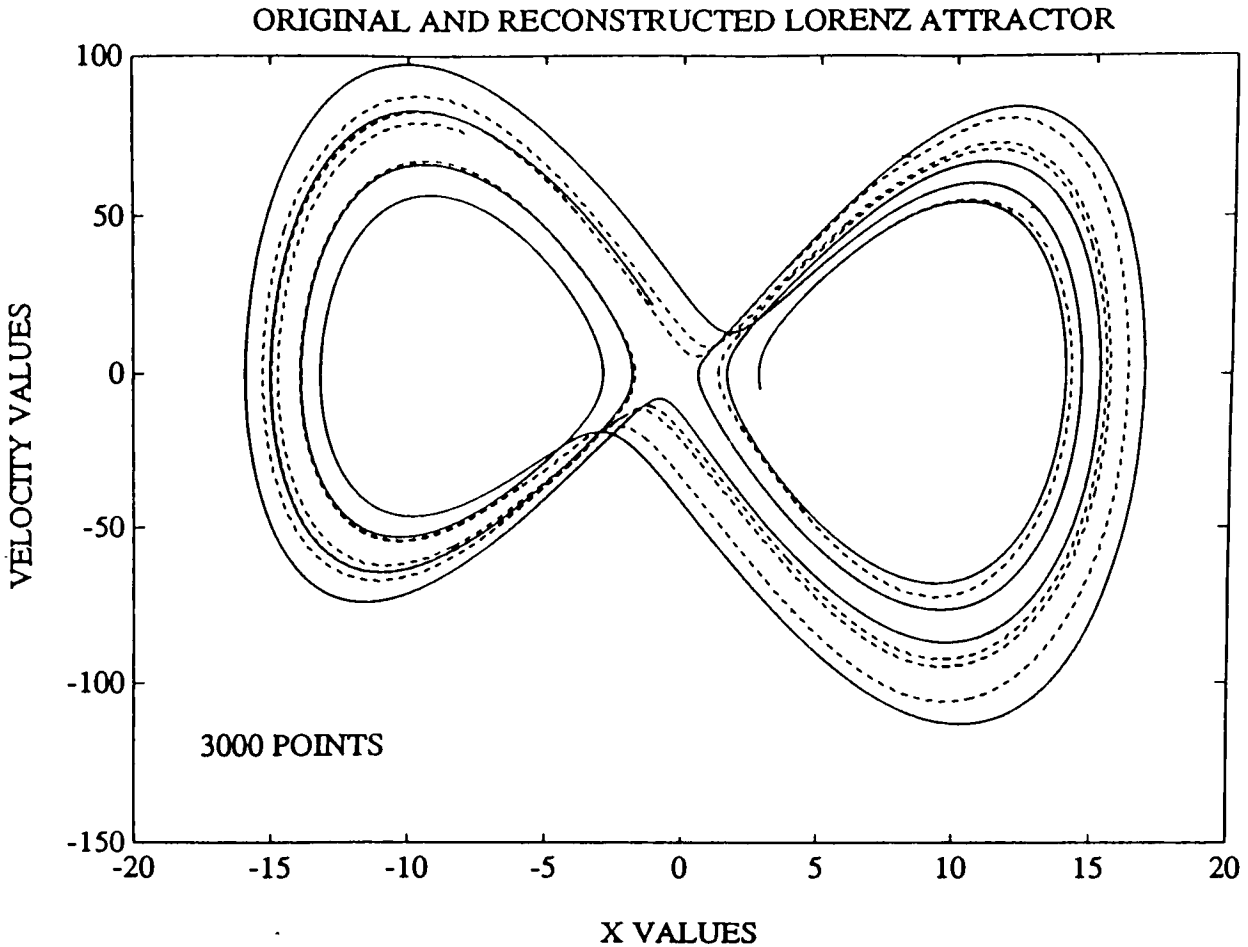
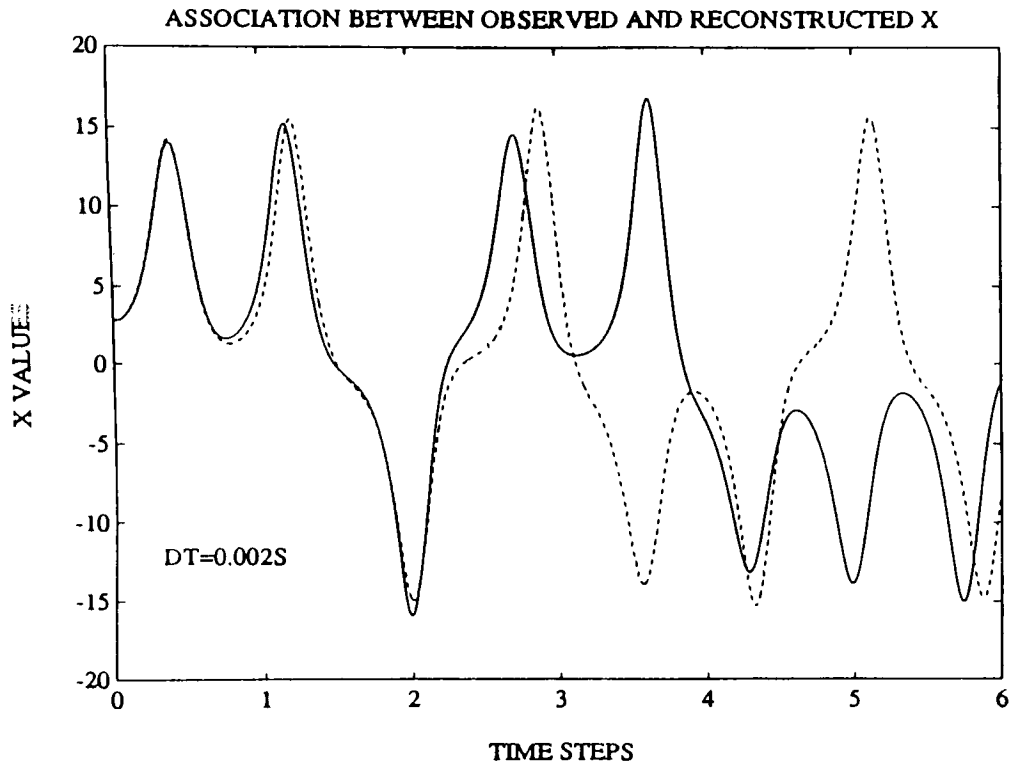


FIGURE 21

This figure shows how good the reconstruction actually is; in this case 3000 points are shown as the further we go, the worse the reconstruction. The reconstructed attractor follows the original for approximately a 1000 points before it breaks away and goes to infinity.



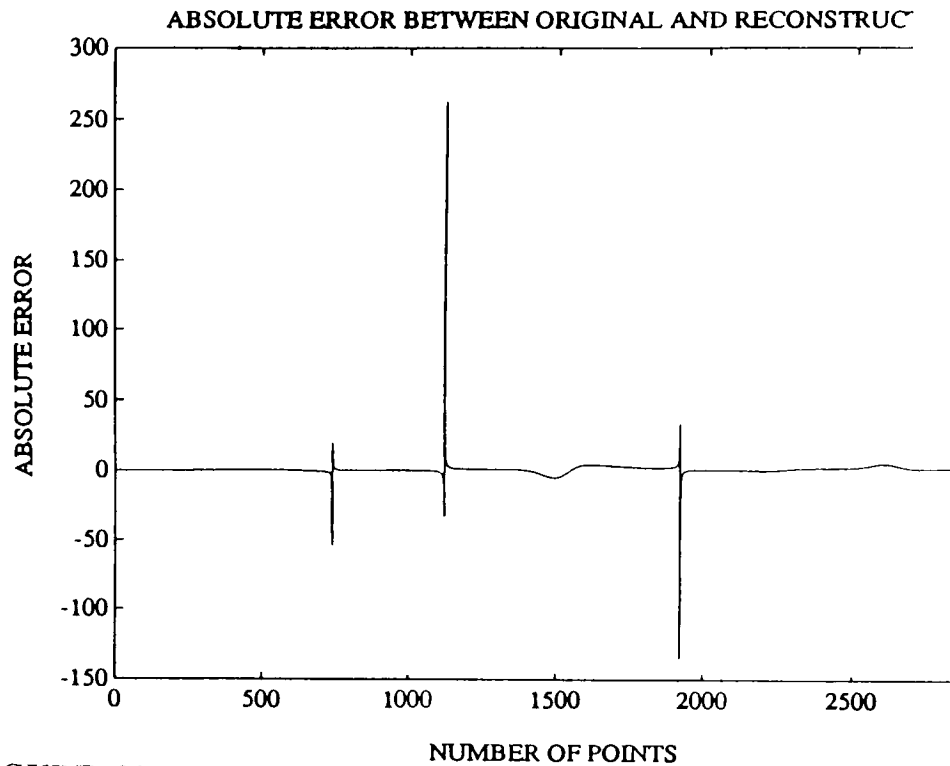


FIGURE 22

(Left to Right) Here once again, we visually get a feel for the difference in the values between the actual x values and the reconstructed x values used to create and recreate the phase plots. As can be seen the values are extremely close for the first 1000 or so points. Also shown is a plot of the absolute error associated with these two variables.

3. PREDICTABILITY OF DYNAMICAL CHAOS

DEGREE OF PREDICTABILITY

One of the central problems of science is forecasting. Given the past, how can we *predict* the future? The classic approach is to build an explanatory model from first principles and measure initial data. Unfortunately this is not always possible. While chaos places a fundamental limit on long term prediction, it suggests possibilities for short term prediction. Random looking data may contain simple deterministic relationships, involving only a few degrees of freedom.

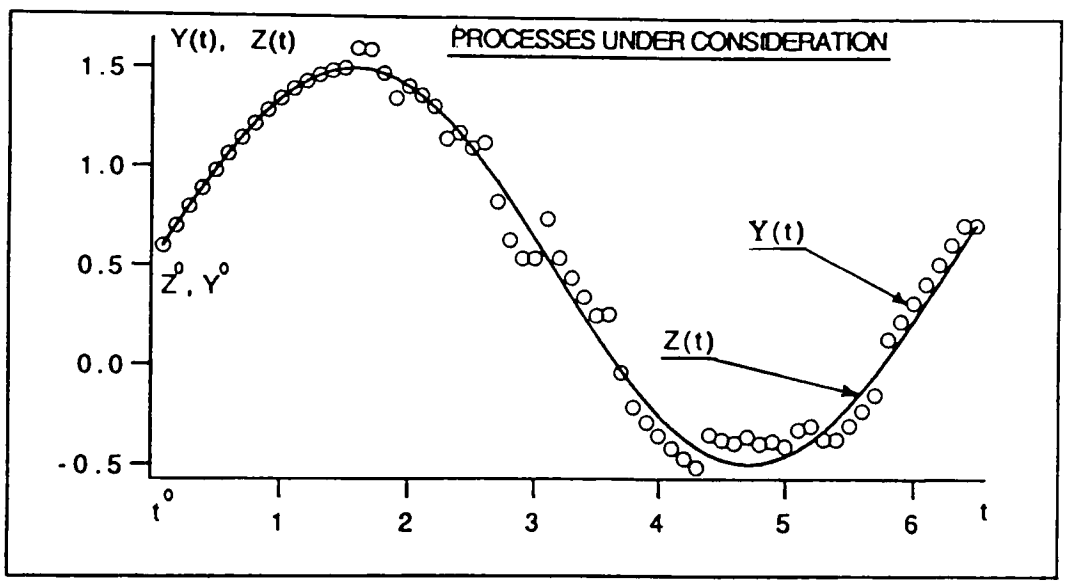
In the previous sections we explored various aspects of nonlinear systems; the process of phase space reconstruction into the future, and the nonlinear models constructed to represent these systems as well as possible. The next step is to establish how good a representation these models are? In other words, how well do they *predict* the future?

Based on the concept of partially deterministic processes, predictability (or unpredictability) is used as a criterion of determinism. In the past, both physicists and mathematicians agreed in handling randomness as the absence of regularity. However, they differed in the treatment of this absence of regularity which appears as the absence of simple algorithms, either as a rapid decay of correlation's, or as a mismatch of observations and predictions.

An important qualitative tool used in measuring the predictability is determining the "Degree of Predictability" or dynamic correlation as used by Kravtsov [1]. Dynamic correlation provides an effective measure of the predictive ability of a

particular dynamical model against a specific data set. The notation D[s] is adopted to indicate the dynamic correlation between a process and an s-step prediction.

First we shall describe the three types of processes involved with regards to forecasting. Consider the following processes:



where by y(t) is an observed process and z(t) a model (predicted) process. The observed process y(t) differs from a real process x(t) (not shown), due to instrumental noise v(t). The process x(t) is a multivariable quantity which represents some physical system and relationship which can be expressed in the following form:

$$K\left(\frac{dx}{dt}, x(t), f(t)\right) = 0$$

where f(t) denotes various fluctuation factors resulting from internal and external sources and K represents an operator. Similarly y(t) takes the form:

$$y(t) = x(t) + v(t).$$

The equation representing z(t) is much simpler as it is constructed from simpler, idealized models of the real process x(t). Prediction is always inferred on the basis of some other model, in other words y(t) is predictable with respect to the specific model process z(t) only. Thus z(t) satisfies

$$K_1 = \left(\frac{dz}{dt}, z(t), 0 \right) = 0.$$

In constructing the model nonlinear system, z(t), there is a freedom in choosing initial conditions, for simplicity, and to explain this point: if $y^o = y(t^o)$ is the initial value of y(t) at time $t = t^o$, then $z^o = z(t^o) = y^o$.

The traditional approach for determining a quantitative measure of the prediction utilized the prognostic error

$$err = (y - z)$$

or rather the variance of this difference. However using Kravtsov's approach, another performance measure was used, namely the Degree of Determinism, D[s];

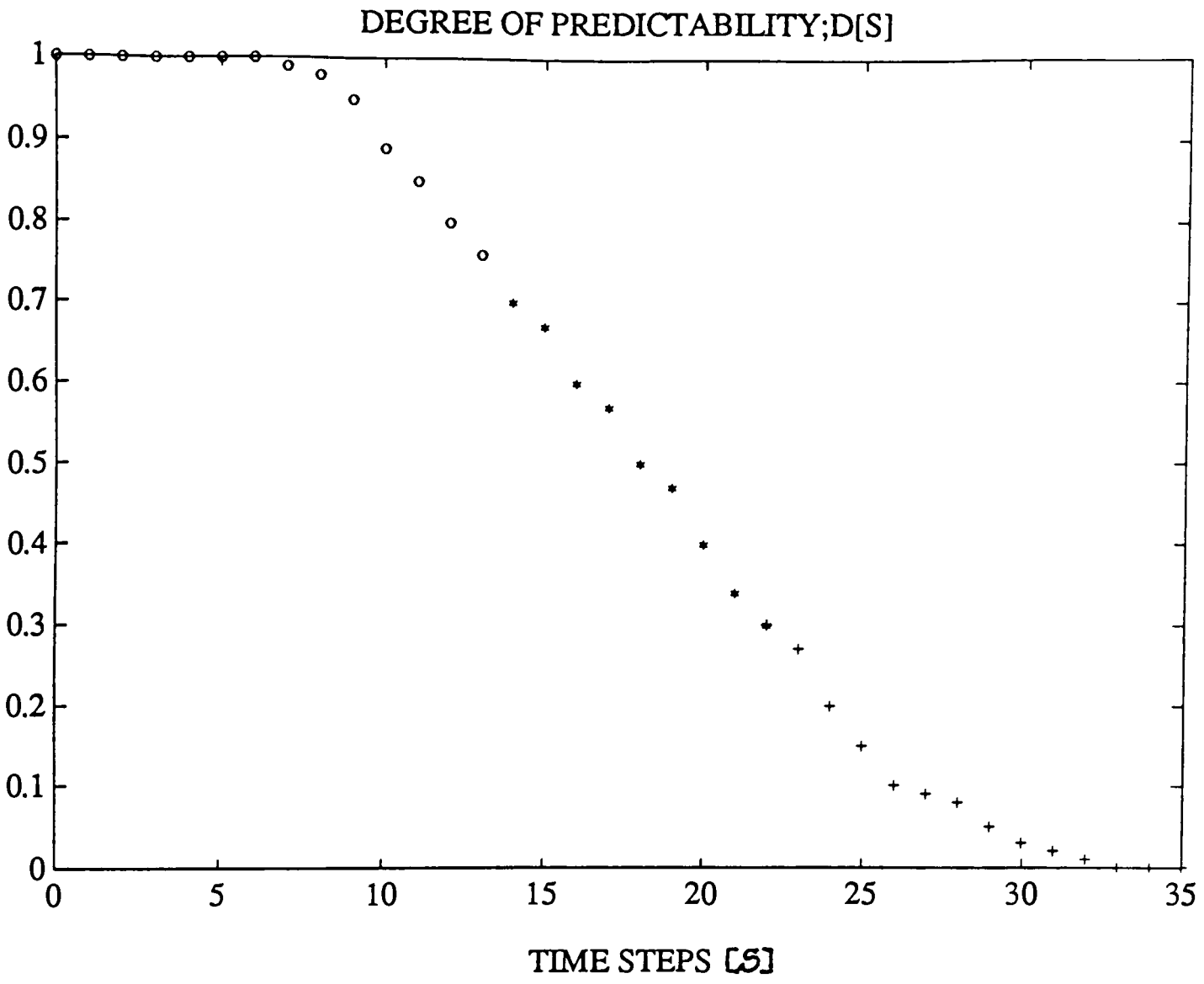
$$D[s] = \frac{\langle y(t)z(t,s) \rangle}{\sqrt{\langle y^2(t)z^2(t,s) \rangle}}$$

If y(t) denotes an observed value of our data set at a time t, then let z(t,s) represent the predicted value of y(t) by using y(t-s) as an initial condition and iterating or integrating the equations of motion s time steps. The angle brackets <.....> represent the variance over all available values of t. We follow Kravtsov, in forming a measurement not just of the one step error variance (i.e. at s=1), but over a wide range of values of s. We form the cross correlation between the observed data and s-step predictions of it . This approach assumes a repeated start-up procedure of the system under study at times t_1^o, t_2^o,....., t_N^o with the subsequent statistical averaging over each obtained realizations.

The absolute value of D[s] is bounded by unity. For those values of s for which D[s] remains close to unity, we can expect to make very accurate predictions with our model, while those values of s which drive D[s] close to zero can be interpreted as defining an extremely unpredictable region. Values of D[s] which lie in between define a region which is partially predictable. An example of a typical D[s] curve is shown on the following page showing the different regions discussed above.

D[s] naturally introduces the time of deterministic behavior, S_{det} , during which the degree of determinability is above some definite level. This level is assumed to be equal to 1/2 i.e. $D[S_{det}] = 1/2$.

Partial determinability of the observed process y(t) with respect to the model process z(t) means that y(t)



TYPICAL FORM OF THE DEPENDENCE OF THE DEGREE OF DETERMINABILITY ON TIME s: (1) o - the region of deterministic behavior; (2) * - the region of partial determinability; (3) + -the region of unpredictable behavior.

(i) is predictable over short intervals of time; where the degree of determinability is rather close to unity, $| 1 - D | << 1$.

(ii) is not predictable at large times; the degree of determinability is small compared to unity, $| D | << 1$.

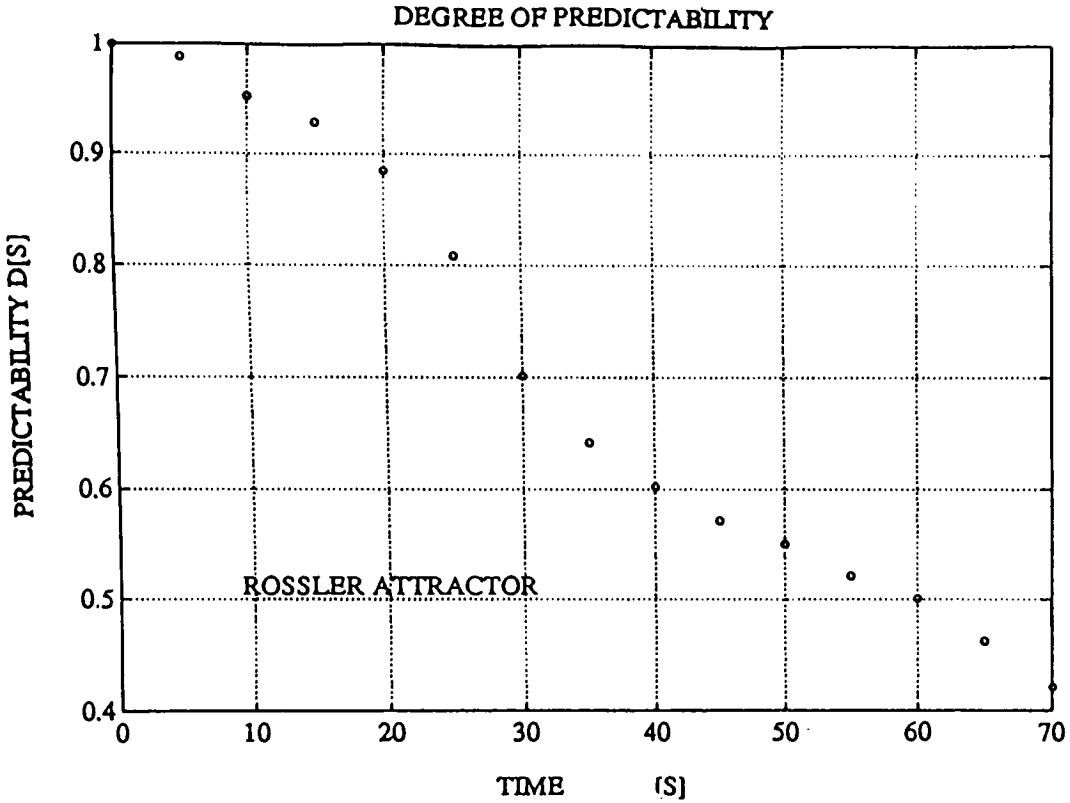
(iii) is partially predictable at over time intervals in the order of S_{det}.

This coefficient of determinability was implemented in Matlab using 'pred2.m'(Appendix 18) to study the two data sets; y being the observed values and z the predicted values. In all simplicity we needed most of all the coefficient matrix that was used for the reconstruction, and then we needed to vary the initial conditions as described earlier. Once again using Matlab 'predict.m' (Appendix 19) was written to essentially take the coefficient matrix (used in the reconstruction) and then using different initial conditions determine the new predicted values.

As is seen from the results (Figures 23 & 24) ,the ODE that describes the Rossler system is fairly good. The coefficient matrix was quite stable. The system was unpredictable after about 60 time steps. The Lorenz system on the other hand, was quite hard to reconstruct it, since so many small time steps were necessary to obtain a fairly good reconstruction. The better the reconstruction obviously the better the prediction qualities. The coefficient matrix determined for this particular system was quite unstable and after very short time spans would cause the reconstructed x values to approach infinity rather quickly. As

can be seen from the results using this approach of reconstruction, the system was unpredictable after approximately 24 time steps.

The method used for predicting nonlinear systems could be said therefore to be quite effective for certain systems. Although it is quite sensitive to the effects of using the optimal time delay it does produce good models that follow the original system for about 1000 points. The only problem however, is that the error associated with the approach tends to accumulate rather quickly thus effecting the predictive abilities that the model provides.



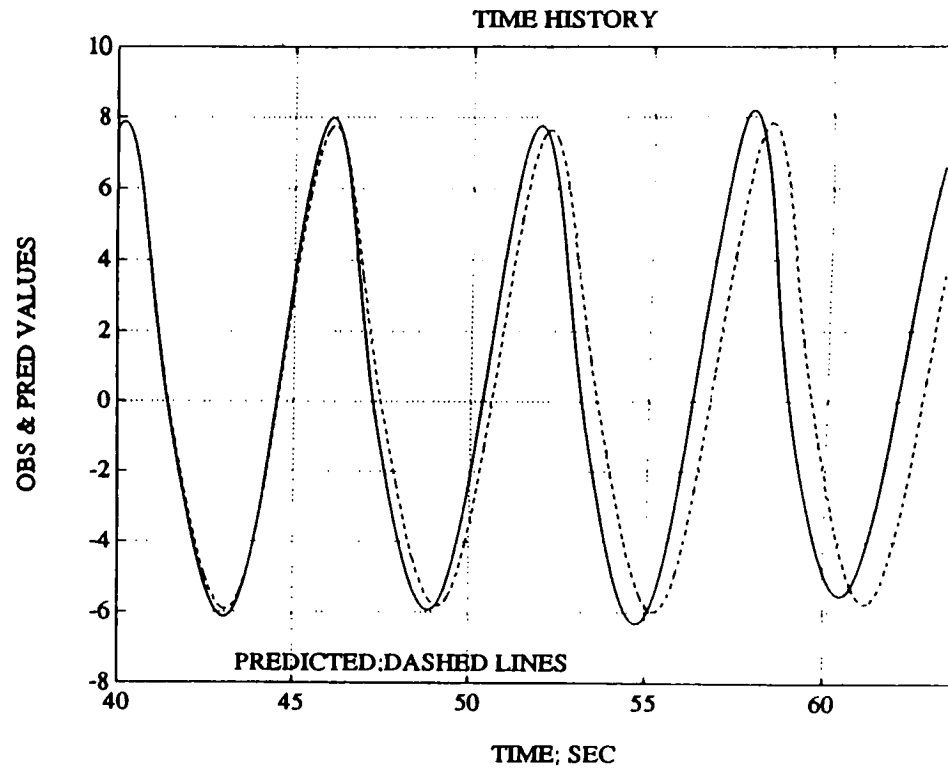
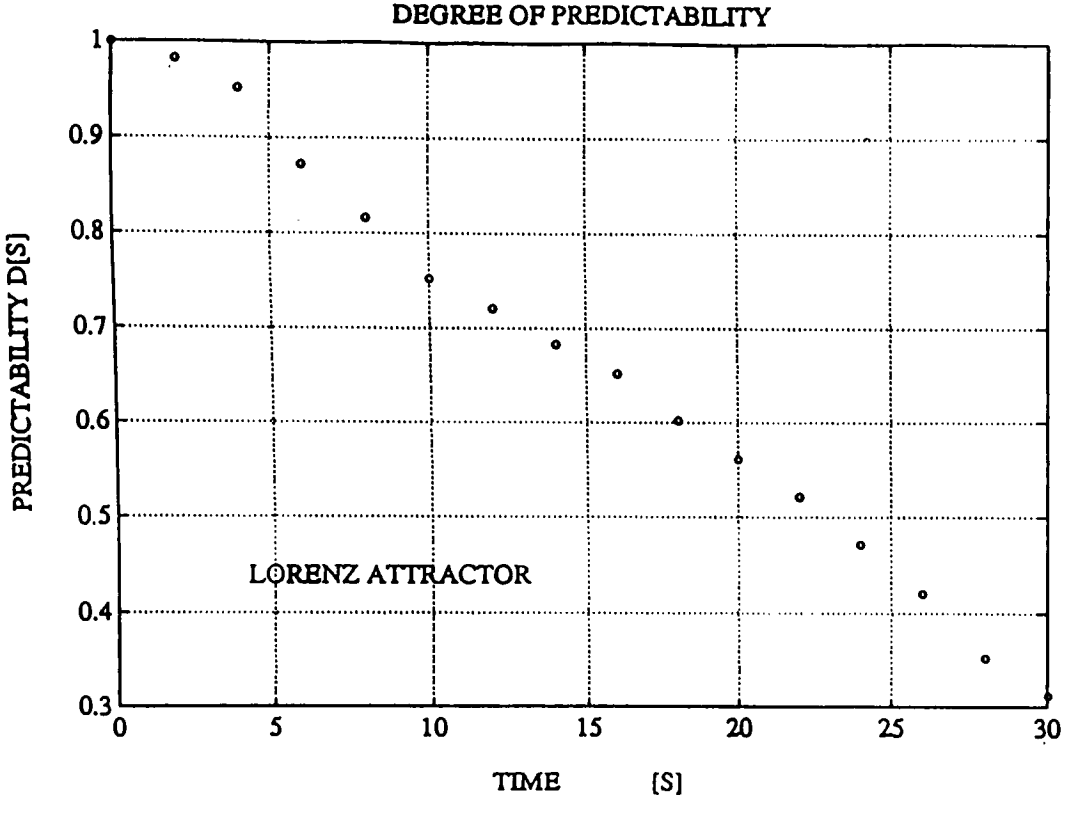


FIGURE 23

These figures show us the predictability of the system, and also gives us a visual feeling as to how good the predictability actually is by comparing the observed and predicted time histories.



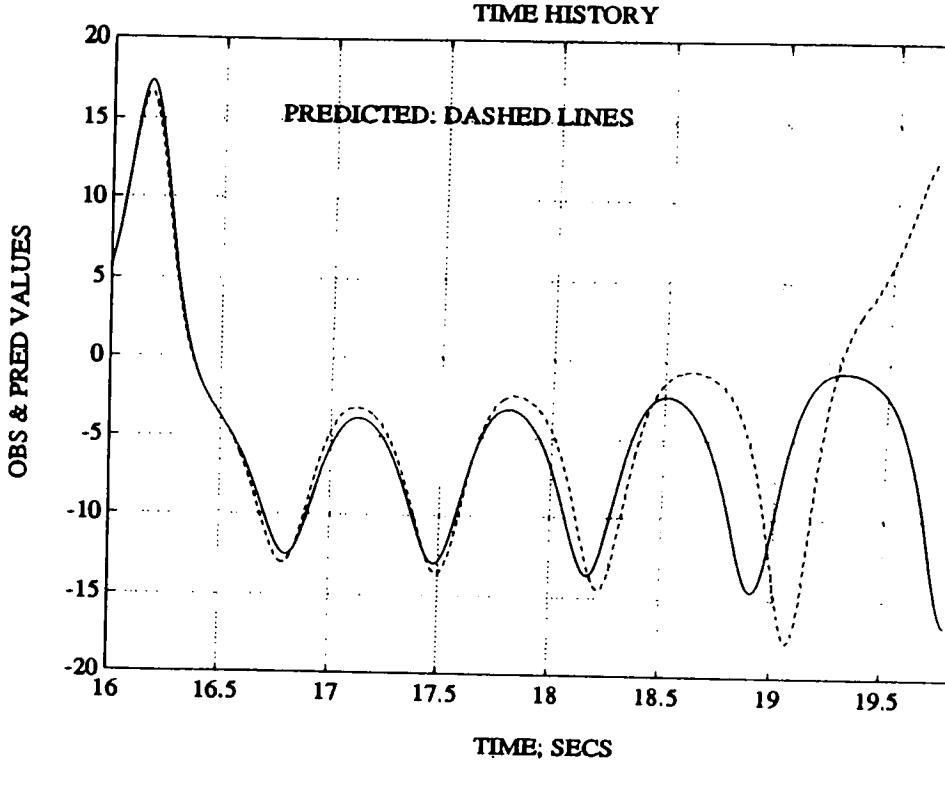


FIGURE 24

Once again, these figures show us the predictability of the system, and also gives us a visual feeling as to how good the predictability actually is by comparing the observed and predicted time histories.

CONCLUSIONS

CONCLUSIONS

The previous sections exposed us to nonlinear system modeling using the Rossler and Lorenz systems as examples. Using each of these systems, we were able to study the effectiveness of the least squares approach in modeling nonlinear systems. The next step was to establish exactly how good a representation these models were able to provide. That is, how well could they predict the future.

The methodology used to construct these models is a rather painstaking process. Let us begin with the determination of the optimal time delay or the optimal shift. As we have seen, this quantity plays an important role in the process, as it allows us to extract either the least, or the most amount of information from the system.

Typically, this value was determined using the minimum value provided by the autocorrelation function, however, this method does not always provide us with an optimal delay value. It was found that the optimal shift occurred at the location of the first inflection point, and if this value was used in the reconstruction of the phase space, we obtained excellent results. This value, however, cannot be assumed to be the optimal value. Other values in its close proximity could prove to be even better. Therefore, much more efficient methods need to be developed so as to obtain a reliable and accurate value.

Determining the embedding dimension also proved to be quite cumbersome. In order to obtain reasonably accurate results using the Grassberger and Proccacia method, we require a large data set (in this case at least 20,000 pts). Usually, in dealing with practical problems, large data sets are not always available and

therefore this method would probably be quite useless. This process is also time consuming, especially if large data sets are used (this also depends on the computer network being used). It should also be noted, that the method is further limited in its application, as it is only accurate for low dimensional systems.

As we have seen, using nonlinear curve fitting we were able to obtain reasonable results. However the process requires a lot of time to run, depending on how many points are being reconstructed. The process, is also quite dependent on the choice of the delay time to be used. Depending on the choice of this value, we could get disastrous results as we saw earlier. The method works well for certain systems, as the Rossler and Lorenz systems would indicate. We were able to construct the Rossler systems extremely well, while the Lorenz proved to be quite a challenge. This might have to do with the fact that the Lorenz system is of a slightly higher dimension than the Rossler system. Thus, the more complex a system, naturally the harder it becomes to study and reconstruct.

The investigation has shown us that it is possible to model nonlinear systems to a certain degree. At the very least, we have learned that even though the term "chaotic", brings to mind "random" and "unpredictable", there is an underlying order, and most of all, there is partial predictability.

APPENDICES

APPENDIX 1

A BRIEF OVERVIEW OF CENTRAL DIFFERENCING

The second degree interpolating polynomial which fits at x_0, x_1, and x_2, this allows us to estimate $f'(x_1)$:

$$f'(x_1) = (1/h)\cdot(\Delta f_0 + (1/2)\cdot\Delta^2\cdot f_0) + \text{error}$$

Writing out difference in terms of f' s:

$$f'(x_1) = (1/h)\cdot(f_1 - f_0 + (1/2)\cdot(f_2 - 2\cdot f_1 + f_0)) = (f_2 - f_0) / 2h + \text{error}$$

The equation above can be called a central - difference approximation. The value of x where it is applied is centered in the range of fit of the polynomial- differences of the function values on either side of $f(x_1)$ are used. Central differencing formulas are decidedly superior in calculating values for derivative. Central- difference formulas can be derived using higher-degree polynomials of even order. Odd-degree polynomials do not have a range of fit that is symmetrical about any of the x-values. For example, the formula corresponding to a fourth-degree polynomial, expressed in terms of function values rather than differences, and related to the point x_0, is

$$f'(x_0) = (1/h)\cdot((f_{-2} - 8f_{-1} + 8f_1 - f_2) / 12$$

In many applications it is preferable to use the simpler formula and control the error by making h small. Observe, however, that the error of the derivative from an n^{th} degree polynomial is of the order h^n, while the error of interpolation is of the order h^{n+1}.

APPENDIX 2

BRIEF OVERVIEW OF LEAST SQUARES

The Least Squares method is based on the criterion is to minimize the sum of the squares of the errors. In addition to giving a unique result for a given set of data, the least squares method is also in accord with the *maximum-likelihood* principle of statistics.

Let Y_i represent an experimental value, and let y_i be a value from the equation:

$$y_i = ax_i + b$$

where x_i is a particular value of the variable assumed free of error. We wish to determine the best values for a and b so that y's predict the function values that correspond to x-values. Let $e_i = Y_i - y_i$. The least-squares criterion requires that:

$$\begin{aligned} S &= e_1^2 + e_2^2 + \ldots\ldots + e_N^2 \\ &= \sum_{i=1}^{N} e_i^2 \\ &= \sum_{i=1}^{N} (Y_i - ax_i - b)^2 \end{aligned}$$

where it is required that S is minimum. N is the number of x, Y pairs. We reach the minimum by proper choice of the parameters a and b, so they are the "variables" of the problem. At a minimum for S, the two partial derivatives $\partial S/\partial a$ and $\partial S/\partial a$ will both be zero. Hence remembering that the x_i and Y_i are data unaffected by our choice of values of a and b, we have

$$\frac{\partial S}{\partial a} = 0 = \sum_{i=1}^{N} 2(Y_i - ax_i - b)(-x_i),$$

$$\frac{\partial S}{\partial b} = 0 = \sum_{i=1}^{N} 2(Y_i - ax_i - b)(-1).$$

Dividing each of these equations by -2 and expanding the summation, we get the solution called *normal equations* :

$$a \sum x_i^2 + b \sum x_i = \sum x_i Y_i ,$$

$$a \sum x_i + bN = \sum Y_i .$$

All summations in the above equation are from i=1 to i=N. Solving these equations simultaneously gives the values for the slope and the intercept and b. Thus if we were to use a larger order polynomial we would follow the same procedure outlined above and so would generate more equations and a bigger matrix to solve. An n^{th} order matrix would be of the following form:

$$\begin{bmatrix} N & \Sigma x_i & \Sigma x_i^2 & \Sigma x_i^3 & \cdots & \Sigma x_i^n \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i^3 & \Sigma x_i^4 & \cdots & \Sigma x_i^{n+1} \\ \Sigma x_i^2 & \Sigma x_i^3 & \Sigma x_i^4 & \Sigma x_i^5 & \cdots & \Sigma x_i^{n+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \Sigma x_i^n & \Sigma x_i^{n+1} & \Sigma x_i^{n+2} & \Sigma x_i^{n+3} & \cdots & \Sigma x_i^{2n} \end{bmatrix} [a] = \begin{bmatrix} \Sigma Y_i \\ \Sigma x_i Y_i \\ \Sigma x_i^2 Y_i \\ \vdots \\ \vdots \\ \Sigma x_i^n Y_i \end{bmatrix}.$$

All the summations in the matrix run from 1 to N.

APPENDIX 3

A BRIEF OVERVIEW OF RUNGE-KUTTA METHODS

A further advance in efficiency(that is, obtaining the most accuracy per unit of computational effort) can be secured with a group of methods due to the German mathematicians Runge and Kutta. The fourth-order Runge-Kutta methods are widely used in computer solutions to differential equations.

To convey some idea of how the Runge-Kutta methods are developed, we show the derivation of a second-order method. We write the increment to y as a weighted average of two estimates of Δy, k_1 and k_2. For the equation $dy/dx = f(x, y)$,

$$y_{n+1} = y_n + ak_1 + bk_2,$$
$$k_1 = hf(x_n, y_n),$$
$$k_2 = hf(x_n + Ah, y_n + Bk_1).$$

We can think of the values k_1 and k_2 as estimates of the change in y when x advances by h because they are the product of the change in x and a value for the slope of the curve, dy/dx. The parameters a,b, A, B are evaluated using techniques not discussed here.

Fourth-order Runge-Kutta methods are most widely used and are derived in a similar fashion. Greater complexity results from having to compare terms through h^4, and gives a set of 11 equations in 13 unknowns. The set of 1 equations can be solved with 2 unknowns being chosen arbitrarily. The most common used set of values leads to the algorithm:

$$y_{n+1} = y_n + (1/6)(k_1 + 2k_2 + 2k_3 + k_4),$$

$$k_1 = hf(x_n, y_n),$$

$$k_2 = hf(x_n + (1/2)h, y_n + (1/2)\,k_1),$$

$$k_3 = hf(x_n + (1/2)h, y_n + (1/2)k_2),$$

$$k_4 = hf(x_n + h, y_n + k_3).$$

The local error term for the fourth-order Runge-Kutta is of the order h^5; the global error would be of the order h^4. It is computationally more efficient than the modified Euler method because, while four evaluations of the function are required per step rather than two, the steps can be manyfold larger for the same accuracy.

APPENDIX 4

```
% A FOURTH ORDER RUNGE-KUTTA ROUTINE TO EVALUATE
%DIFFERENTIAL EQUATIONS.
% function [q,t]=rk4(f,ti,dt,tf,q0);

function [q,t]=rk4(f,ti,dt,tf,q0);

% MATRIX SIZING
t=0:dt:tf;
nf=(tf-ti)/dt;
wd=length(q0);
q=NaN*ones(nf,wd);
q(1,:)=q0;
for n=1:1:nf
   k1 = dt*feval(f, q(n,:), t(n));
   k2 = dt*feval(f, q(n,:) + 0.5*k1 , t(n) + 0.5*dt);
   k3 = dt*feval(f, q(n,:) + 0.5*k2 , t(n) + 0.5*dt);
   k4 = dt*feval(f, q(n,:) + k3 , t(n) + dt);

   q(n+1,:)=q(n,:)+(1/6)*(k1 + 2*k2 + 2*k3 + k4);
end;
```

APPENDIX 5

```
%     This is a MATLAB program to be called by RK4.M to
%     evaluate the x, y and z values of the  ROSSLER attractor.
%
%           [qprime]=rossler(q,t)

function [qprime]=rossler(q,t)
%     PARAMETERS
a=0.2;b=0.2;c=4.6;
%     FUNCTIONS
qprime=[-q(3)-q(2),q(1)+a*q(2),b+q(3).*(q(1)-c)];
```

APPENDIX 6

```
%       This is a MATLAB program to be called by RK4.M to evaluate the x,y and z
%       values of the LORENZ attractor
%
%                       [qdot]=lorenz(q)

function [qdot]=lorvel(q)
%       PARAMETERS
a=10;b=8/3;c=28;
%       FUNCTIONS
qdot(:,1)= -a*q(:,1)+a*q(:,2);
qdot(:,2)=c*q(:,1)-q(:,2)-q(:,1).*q(:,3);
qdot(:,3)=-b*q(:,3)+q(:,1).*q(:,2);
```

APPENDIX 7

```
%     This is a MATLAB program to evaluate the velocities of a
%     ROSSLER attractor after the x, y and z values have been generated
%     using the RK4.M and ROSSLER.M programs.
%
%          [qdot]=rosvel(q)

function [qdot]=rosvel(q)
%    PARAMETERS
a=0.2;b=0.2;c=4.6;
%    FUNCTIONS
qdot(:,1)= -q(:,3)-q(:,2);
qdot(:,2)= q(:,1)+a*q(:,2);
qdot(:,3)= b+q(:,3).*(q(:,1)-c);
```

APPENDIX 8

```
%     This is a MATLAB program to evaluate the velocities of a
%     LORENZ attractor after the x, y, and z values have been  generated
%     using the RK4.M and LORENZ.M programs.
%
%                [qdot]=lorvel(q)

function [qdot]=lorvel(q)
%     PARAMETERS
a=10;b=8/3;c=28;
%     FUNCTIONS
qdot(:,1)= -a*q(:,1)+a*q(:,2);
qdot(:,2)=c*q(:,1)-q(:,2)-q(:,1).*q(:,3);
qdot(:,3)=-b*q(:,3)+q(:,1).*q(:,2);
```

APPENDIX 9

```
%  THIS IS A MATLAB PROGRAM TO EVALUATE THE OPTIMUM SHIFT FOR
%  A GIVEN TIME SERIES, x.
%
%                  function [k1,ac]=autocorr(x)

function [k1,ac]=autocorr(x)
disp('Calculating the Delay Step')

%  ARRAY SIZING
n=length(x);
n2=fix(n /10);

%  CALCULATE AUTO CORRELATION
ac=ones(1,n2);
for k=1:n2;
   ac(k)=sum(x(1:n-k+1).*x(k:n))/(n-k+1);
end;

len=length(ac);
%  FIND VALUE WHERE MINIMUM AUTO CORRELATION VALUE OCCURS

 ac1=ac(1:len-1);
 ac2=ac(2:len);
 i=find(ac1-ac2<0);
k1=ceil(i(1)-1);
disp(['The first local minimum occurs at ', num2str(k1)])
```

APPENDIX 10

```
%  THIS IS A MATLAB PROGRAM TO DETERMINE THE OPTIMAL SHIFT FOR
%  A GIIVEN TIME SERIES, x.
%
%                [k1,dc]=decorr(x,ndim,dt)

% ndim: represents the number of dimensions,
% dt : the time step
% k1: autocorrelation value

function [k1,dc]=decorr(x,ndim,dt)

%  INITIAL ARRAY SIZING
len=length(x);
n2=fix(len /10);

%  CALCULATE SHIFT
dc=ones(1,n2);
for j=1:n2;
   n=len-(ndim-1)*j;
   X=x(1:n+1);
   for k=1:ndim-1
        X=X.*x(k*j:n+k*j);
   end
   dc(j)=sum(X)/(n+1);
end;
dc=dc';
size(dc);

%  FIND DECORRELATION

[dcshort,dcdot]=cent2(dc,dt);
len=length(dcdot);
if dcdot(1) >= dcdot(2)
   i=find(dcdot(1:len-1) <= dcdot(2:len));
```

```
else
   i=find(dcdot(1:len-1) >= dcdot(2:len));
end
k1=i(1) + 1;
disp(['The first inflection occurs at ', num2str(k1),...
     ' for ', num2str(ndim),' dimensions. '])
```

APPENDIX 11

```
%       This is a MATLAB program that determines the dimensionality of a system
%        using the Grassberger and Procaccia approach.

%              [D,N,Ns,a,dAc]=dimembed(x,dt)

function [D,N,Ns,a,s,dAc]=dimembed(x,dt)

%  INITIALIZING VARIABLES .
clock1=clock;
x=x(:);
pmax=9;
ndim=3; skip=2; s1=1; s2=0; c=0;
Rmin=log(1e-13);Rmax=log(1e+13);
clg;
while s1 > (ndim-9)
 if c~=1
%  FIND OPTIMAL DELAY VALUE FOR EMBEDDING DIMENSION
  [dAc,dc]=decorr(x,ndim,dt);
   subplot(211),plot(dc);

%  CREATE DELAYED COORDINATE VECTOR SPACE
   len=length(x)-(ndim-1)*dAc;
   N2=len*(len-1)/2;

   xd=ones(len,ndim);
  for p=1:ndim;
      xd(1:len,p)=x((p-1)*dAc+1:(p-1)*dAc+len);
  end;
 end
%  GENERATE RADIUS INCREMENTS AND ALLOCATE N-VECTOR SIZE
  Rmin=0.7*Rmin+0.3*Rmax;
  Rmax=0.5*Rmin+0.5*Rmax;
  if Rmax > 0
    Rmax =0;
```

```
    for p=1:pmax
     n(p)=sum(r <= R1(p));
     N1(p)=N1(p) + n(p);
    end;
   end;
 end;
 if c > 0
  D(:,c)=R1;
  N(:,c)=log(N1 ./N2);
  dAc(c,1:2)=[ndim,dAc];
clg;
%   CALCULATE SLOPE AND CONSTRUCT LINES
   a(c,:)=([D(:,c),ones(D(:,c))]\N(:,c))';
   Ns(:,c)=[D(:,c),ones(D(:,c))]*a(c,:)';
   subplot(212),plot(D(:,c),N(:,c),'or',D(:,c),Ns(:,c),'-b'),...
   pause(1);

   s2=s1;s1=a(c,1);
   disp(['The Slope for a ',num2str(ndim),...'
     ' dimensional embedding is ',num2str(a(c,1))])
   disp([num2str(etime(clock,clock1)/60),' minutes have elapsed'])
   disp([' '])
  end
  c=c+1;
  ndim=ndim+skip;
end

%   CONSTRUCT LINES DEPICTING SLOPES
clg;
plot(D,N,'or',D,Ns,'-b'),grid;
```

```
 end
 dr=(Rmax-Rmin) /(pmax-1);
 R1=(Rmin:dr:Rmax)';
 N1=zeros(pmax,1);n=zeros(pmax,1);

%  RESET MINIMUM AND MAXIMUM RADII AND CHECK COUNTER
 Rmax=log(1e-13);Rmin=log(1e+13);
 if c<=0
    disp(['Calculating initial maximum and minimum radii for '...
      ,num2str(ndim),' dimensions.'])
    ndim=ndim-skip;
 else
    disp(['Calculating Slope for a ',num2str(ndim),...'
    ' dimensional embeding.'])
 end

%  CALCULATE DISTANCES BETWEEN  ith AND jth ROWS
 for k=1:len-1;
    k1=len-k;
    p=(k)*ones(1,k1);
    xn=xd(p,:)-xd(k+1:len,:);
    r= 0.5*log(sum((xn.^2)'));

%  SEARCH FOR MINIMUM AND MAXIMUM RADIUS FOR NEXT ITERATION
    rmax=max(r);
    if rmax > Rmax
     Rmax=rmax;
    end;

    rmin=min(r);
    if rmin < Rmin
     Rmin=rmin;
    end;

  if c > 0
%  COUNT NUMBER OF POINTS IN EACH RADIUS INCREAMENT
```

APPENDIX 12

```
% This program fits a polynomial to X-DOT using CENTRAL DIFFERENCING !
% Derivatives at the two endpoints are not estimated.
%  [coef2]=cent6(x,dt)

function [coef2,xdot]=cent6(x,dt);

[NP,MP]=size(x);
%
xdot=zeros(NP-6,MP);
xdot(1:NP-6,:)=(x(7:NP,:)-9*x(6:NP-1,:)+45*x(5:NP-2,:)-45*x(3:NP-4,:)+...
        9*x(2:NP-5,:)-x(1:NP-6,:)) ./(60*dt);

%   CALL NLFIT.M
coef2=nlfit(x(4:NP-3,:),xdot);
%
```

APPENDIX 13

```
%       This Matlab program is called by CENT6.M to use the derivative values
%       generated for a nonlinear curve fit. The coefficients are generated using
%       least squares.

% [a]=nlfit(x,xd)

function [a]=nlfit(x,xd)

% INITIAL MATRIX SIZING
[m,n]=size(x);
[p,q]=size(xd);
c=0;
x1=ones(m,1);
x=[x1 x];

% NONLINEAR GENERATION
for k1=1:n+1
   for k2=k1:n+1
      for k3=k2:n+1
         for k4=k3:n+1
               for k5=k4:n+1
 c=c+1;
 X(:,c)=x(:,k1).*x(:,k2).*x(:,k3).*x(:,k4).*x(:,k5);
           end
           end
        end
   end
end

% DETERMINATION OF COEFFICIENT MATRIX (a)
for k=1:n
   a(k,:)=(X\xd(:,k)).';
end
```

APPENDIX 14

```
%       THIS MATLAB PROGRAM IS CALLED BY RKPOLY.M TO EVALUATE
%       THE DERIVATIVE VALUES FOR NEW RECONSTRUCTED VALUES OF
%       X USING NONLINEAR GENERATION AND THE COEFFICIENT MATRIX.

function [qd]=nlpoly(q,a)

%       MATRIX SIZING
n=length(q);
c=0;
q1=[1 q];

% NONLINEAR GENERATION
for k1=1:n+1
   for k2=k1:n+1
      for k3=k2:n+1
         for k4=k3:n+1
            for k5=k4:n+1
c=c+1;
Q(1,c)=q1(:,k1).*q1(:,k2).*q1(:,k3).*q1(:,k4).*q1(:,k5);
                end
            end
         end
   end
end

% VELOCITY EVALUATION USING COEFFICIENT MATRIX
qd=a*Q';
qd=qd';
```

APPENDIX 15

```
%    THIS MATLAB PROGRAM EVALUATES THE FUNCTION GENERATED BY
%    NLPOLY.M USING A FOURTH ORDER RUNGE-KUTTA ROUTINE.
%     function [q,t]=rkpoly(a,q0,dt,m);

function [q,t]=rkpoly(a,q0,dt,m);
f='nlpoly';

% MATRIX SIZING
n=length(q0);
q=NaN*ones(m,n);
q(1,:)=q0;
diff=0;
for n=1:1:m-1
   k1 = dt*feval(f, q(n,:), a);
   k2 = dt*feval(f, q(n,:) + 0.5*k1 , a);
   k3 = dt*feval(f, q(n,:) + 0.5*k2 , a);
   k4 = dt*feval(f, q(n,:) + k3 , a);

   q(n+1,:)=q(n,:)+(1/6)*(k1 + 2*k2 + 2*k3 + k4);
end;
```

APPENDIX 16

```
%       THIS MATLAB PROGRAM  EVALUATES THE VELOCITIES USING THE
%       RECONSTRUCTED X VALUES

function [qd]=nlpvel(q,a)

% MATRIX SIZING
[m,n]=size(q);
c=0;
uno=ones(m,1);
q1=[uno q];

% NONLINEAR GENERATION
for k1=1:n+1
   for k2=k1:n+1
      for k3=k2:n+1
         for k4=k3:n+1
                for k5=k4:n+1
c=c+1;
Q(:,c)=q1(:,k1).*q1(:,k2).*q1(:,k3).*q1(:,k4).*q1(:,k5);
            end
            end
        end
   end
end

% VELOCITY EVALUATION USING COEFFICIENT MATRIX
qd=a*Q';
qd=qd';
```

```
%       This is a MATLAB program written to encapsualte all the programs at
%       once. It's basic function is to reconstruct a given time series as well as
%       possible.

%                       [coef,q,qd]=recon(X,dt,delay,dim,points)

function [coef,q,qd]=recon(X,dt,delay,dim,points)

%     MATRIX SIZING
len=length(X);

%  Generate delayed vector space

for k=1:dim
   Xd(:,k)=X(1+(k-1)*delay:len+(k-dim)*delay);
end

%  Generate nonlinear equations to describe motion
[coef,xd]=cent6(Xd,dt);
[q,t]=rkpoly(coef,Xd(4,:),dt,points);
qd=nlpvel(q,coef);
```

APPENDIX 18

```
% THIS PROGRAM COMPUTES THE PREDICTABLITY OF A GIVEN
% NON-LINEAR SYSTEM.

% function[D,S]=pred(z,y)

function[D,S]=pred2(z,y)

N=length(y);
y=y(:,1);
z=z(:,1);
m=1;
%CALCULATE DETERMINABILITY
SSTO=sum(y.^2)-(sum(y)^2)/N;
SSR1=(sum(y.*z)-((sum(z)*sum(y))/N))^2;
SSR2=sum(z.^2)-(sum(z))^2/N;
SSR=(SSR1)/(SSR2);
R=(SSR)/(SSTO);
D(m,:)=R;
S(m,:)=m;
end;

%Plot Graph of Determinability
i1=(D > 0.5);d1=D(i1);
i2=(D==0.5);d2=D(i2);
i3=(D < 0.5);d3=D(i3);
%plot(t,d1,'o',t,d2,'*',t,d3,'+');
plot(S,D,'o');
title('DEGREE OF PREDICTABLITY');
xlabel('TIME STEP [s]');
ylabel('D[s]');
```

APPENDIX 19

```
%     This is a MATLAB program to evaluate  a new time series based on a
%     given initial condition. It is used once the coefficient matrix has been
%     determined.
%
%                [q,qd]=predict(coef,x,start,delay,dt,points)

function [q,qd]=predict(coef,x,start,delay,dt,points)

% Matrix Sizing
[m,n]=size(x);
[dim,q]=size(coef);

%  Generate delayed vector space
for k=1:dim
  x0(1,k)=x(start + (k-1)*delay);
end

%  Evaluate based on delayed vector
[q,t]=rkpoly(coef,x0,dt,points);
qd=nlpvel(q,coef);
```

APPENDIX 20

```
%       This is a MATLAB program to determine a good coefficient matrix for a
%       given time series. It basically shows the error related to the initial
%       conditions.

%                   [coef,X,Xd]=qdot(x,dt,delay,dim)

function [coef,X,Xd]=qdot(x,dt,delay,dim)

%       MATRIX SIZING
len=length(x);

%  Generate delayed vector space

for k=1:dim
   X(:,k)=x(1+(k-1)*delay:len+(k-dim)*delay);
end

%  Generate nonlinear equations to describe motion
[coef,xd]=cent6(X,dt);
Xd=nlpvel(X,coef);
```

REFERENCES

REFERENCES

1. Yu. A. Kravtsov; Randomness and Predictability in Dynamic Chaos. Nonlinear Waves, Dynamics and Evolution. Vol. 2 (Research reports in Physics, pgs 44-56)Berlin; New York, 1989.

2. A. A. Tsonis & J.B. Elsner; Nonlinear prediction as a way of distinguishing chaos from random fractal sequences. Letters to nature; Vol 358, 16 July 1992.

3. Jeffrey S. Brush & James B. Kadtke; Model requirements fro NLD based noise reduction. RTA technical reports, RTA Corporation, Springfield, VA, 1991.

4. Jeffrey S. Brush & James B. Kadtke; Nonlinear signal processing using empirical global dynamical equations. RTA technical reports, RTA Corporation, Springfield, VA, 1992.

5. W. Lauterborn & U. Parlitz; Methods of Chaos Physics and their applications in acoutics. J. Acoust. Society of America, Vol. 84, No.6 Dec 1988.

6. A.G. Darbyshire & T.J. Price; Phase Portraits from Chaotic Time Series, pgs 247-257, 1989.

7. J. Doyne Farmer & John J. Sidorowich; Predicting Chaotic Time Series. Physical Review Letters. Vol. 59, No. 8, 24 Aug 1987.

8. N.H. Packard, J.P. Crutchfield, J.D. Farmer & R.S. Shaw; Geometry from a Time Series. Physical Review Letters, Vol. 45, No. 9, 1 Sep 1980.

9. Peter Grassberger; Do cilmatic attractors exist? Letters to Nature, Vol. 323, 16 Oct 1986.

10. Christopher Essex, Turab Lookman & M.A.H. Nerenberg;
The climate attractor over short time scales. Letters to Nature, Vol. 326, 5 March 1987.

11. A.A. Tsonis & J.B. Elsner; The weather attractor over very short time scales. Letters to Nature, Vol. 333, 9 June 1988.

12. Henry D.I. Abarbanel; Prediction in Chaotic Nonlinear Systems: Methods for Time series with broadband Fourier spectra. Physical Review Letters, Vol. 41, No. 4, 15 Sep 1990.

13. Richard L. Smith; Estimating Dimension in Noisy Chaotic Time Series. Royal Statistical Society, Vol. 54, No.2, pg 329-351, 1992.

14. J. Jimenez, J.A. Moreno, G.J. Ruggeri; Forecasting on a Chaotic time series: A local optimal linear reconstruction method. Physical Review Letters, Vol. 45, No. 6, 15 March 1992.

15. Peter Grassberger & Itamar Procaccia; Characterization of Strange Attractors. Physical Review Letters, Vol. 50, No. 5, 31 Jun 1983.